DIGITIZATION OF AN EDUCATIONAL ZEN BUDDHIST BOARD GAME

by

NITHI DEIVANAYAGAM

A THESIS

Presented to the Department of Computer Science
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

April 2025

An Abstract of the Thesis of

Nithi Deivanayagam for the degree of Bachelor of Science
in the Department of Computer Science taken April 2025

Title: Digitization of an Educational Zen Buddhist Board Game

Approved: *Allen Malony, Ph.D.*
Primary Thesis Advisor

This thesis describes the design and implementation of a computer-based educational game for Zen Buddhist training and practice, called the Training & Enlightenment e-Game (TAEeG). TAEeG is a digital version of the Training & Enlightenment© (TAE) board game, which was manufactured by the Order of Buddhist Contemplatives (OBC) in the 1980s. TAEeG software was developed using Python, together with the Tkinter library, to create the graphical user interface (GUI). The GUI preserves key features of the original game board, including a 2-dimensional Wheel of Training (*Dharmachakra*) and presentation of brief ethical narratives, to represent core teachings in Soto Zen Buddhism and to provide opportunities for meditation and application of Buddhist precepts in real-life scenarios. Software development includes database management, algorithmic progression, pseudorandomized movement of player avatars, and interactions between the Database, Game, and GUI Modules. This project was completed under the guidance of Prof. Allen Malony (Computer and Information Sciences) and Prof. Gwen Frishkoff (Cognitive and Learning Sciences). The software code (TAEeG game mechanics) is licensed under the GNU General Public License (GPLv3). The database (TAE game contents) is copyrighted, together with the original TAE board game.

Acknowledgements

In this section, I would like to sincerely thank all the amazing people and organizations who have contributed to the completion of this thesis. I express utmost gratitude to my two advisors, Dr. Allen Malony and Dr. Gwen Frishkoff for their knowledge, expertise, and constant support during this journey.

I am grateful to the many monastic and lay practitioners of Soto Zen, whose laid the foundation for this project. These sources have inspired me with their profound insights and persistent dedication to spiritual growth. I thank Shasta Abbey and the Order of Buddhist Contemplatives (OBC) for extending permission to work on this project. Reverend Master Ando Mueller at Shasta Abbey deserves special thanks for introducing us to the original Training and Enlightenment (TAE) board game. Reverend Master Hugh Gould (Prior, Eugene Buddhist Priory) explained the history and key Buddhist principles behind the TAE game (see Interview transcription in Section 2) and introduced me to Zen Buddhist meditation.

I thank a contributor (who wishes to remain anonymous) for assisting with code development. I also thank Dr. Anthony Hornof for sharing his expertise in a Software Methodology course, which taught me valuable techniques that informed software development.

A special thanks to my classmates in HC 477 (with Professor Anita Chari), who created an environment that stimulated critical thinking, participated in interesting conversations, and offered constructive feedback.

This acknowledgment is a token of my sincere gratitude for my family's and friends' constant support. Their support and understanding have been a consistent source of energy during this academic journey. My efforts have been motivated by their determination and confidence in the importance of this work.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Motivation and Aims

The Training & Enlightenment e-Game (TAEeG) is a digital adaptation of the Training & Enlightenment© (TAE) board game, which was created by Reverend Master Jiyu-Kennett, founder of the Order of Buddhist Contemplatives (OBC). Approximately 300 games were manufactured and distributed by the OBC in the 1980s and continue to be used in Buddhist temples across the U.S., U.K., Canada, and Europe. The present project aims to preserve this educational tool and to extend its reach to future generations. It uses the Tkinter library to create a Python-based game board system with a graphical user interface (GUI), algorithmic coding, and error handling during code execution. To the best of our knowledge, TAEeG is the first computer game that is designed to support Zen Buddhist training and practice.

1.1.1. Dharmachakra: The Buddhist Wheel of Training

The TAE game board features a central wheel, with three circular tracks and a central hub, which are connected by eight spokes, as shown in Figure 1.

(A) (B)

Figure 1. Dharmachakra (Wheel of Training).
(*A*) Schematic Representation of Wheel. (*B*) Eight Spokes, representing *Noble Eightfold Path*.

The Wheel of Training (*Dharmachakra*) is an ancient symbol and is used to convey several key teachings, including the Four Noble Truths and the Noble Eightfold Path.

Four Noble Truths. In the Buddha's first teaching to his disciples after his Enlightenment, he is said to have laid out the Four Noble Truths (Jiyu-Kennett, 1999: pp. 8–13):

1. **The Problem (*dukkha*)**: All beings experience suffering (Skt. *dukkha*). One goal of Buddhist practice is to understand and accept that suffering exists in a variety of forms. In TAE, players come to experience how *dukkha* arises as they make their way around the Wheel of Training. "Turning the wheel" is a metaphor for skillful practice: once players enter the Wheel, they move in a clockwise, concentric manner — from the outer spokes to the inner hub — representing movement towards liberation from *dukkha*.

2. **The Source of the Problem (*desire*)**: A second goal of practice is to realize that suffering arises when we do not want what we have (the state of *aversion* — e.g., anger, anxiety, or disgust), or when we desire what we do not have (the state of craving or *attachment*). In TAE, players are invited to reflect on their state of mind with the arising of different wordly conditions (favorable or unfavorable game-based scenarios).

3. **The Resolution of the Problem (*nirvana*)**: A third goal is to see that letting go of desire releases us from suffering. In TAE, liberation is symbolically indicated by reaching the center (hub) of the wheel, symbolizing *kensho* (Jp. "glimpse of *nirvana*") and in the progression through different avatar ranks, representing spiritual maturation (cf. section below on Wisdom).

4. **The Way to the Resolution (*Noble Eightfold Path*)**: According to Buddhism, the way to resolve *dukkha* is to practice the Eightfold Path, a.k.a. the three "pillars" of practice: (1) **Ethics** (Skt. *Sila*), (2) **Meditation** (Skt. *Samadhi*), and (3) **Wisdom or Insight** (Skt. *Prajna*).

(1) ***Ethics.*** The first pillar of practice is Ethics (Skt. *Sila*), which includes Speech, Action, and Livelihood. At the start of the TAE game, players study and commit to a set of ethical intentions, known as the Buddhist Precepts (Appendix A: Table 8). The precepts clarify what is meant by "right" (or wise) conduct of body, speech, and mind. Karmic narratives are presented throughout the game and present opportunities to understand and practice the precepts.

Table 1 shows sample narratives in TAE. Each narrative is divided into three parts: (1) Karmic context (situation); (2) Karmic action (Skt. ***karma***); and (3) Karmic consequence (Skt. ***vipaka***). Comments indicate whether each action is skillful or "positive" (i.e., aligned with precepts), or unskillful or "negative" (misaligned with precepts). Note that "action" refers to internal functions of the mind (i.e., feelings and thoughts), as well as observable functions of the body (i.e., speech and bodily actions). Game-based consequences (e.g., "lose a turn," "increase avatar rank," "go to center") are explained in Section 2.2.

Table 1. Sample Karmic Narratives, illustrating "good" (**skillful**) and "bad" (**unskillful**) actions of Body, Speech, & Mind (Jiyu-Kennett, 1989). See "Comment" for associated Precept (ethical principle).

Karmic Context	Karmic Action (*Karma*)	Karmic Consequence (*Vipaka*)	Comment (Type of Karma)
You're at a party.	You drink too much and become intoxicated.	Lose one turn.	Action of ***Body*** (–) (Great Precept #5)
You're on the telephone.	You participate in gossip.	Lose two turns.	Action of ***Speech*** (–) (Great Precept #6)
Your name is misspelled.	You get very angry.	Go to Hell.	Action of ***Mind*** (–) (Great Precept #9)
You remove your shoes.	You put your shoes straight.	Increase your Avatar Rank.	Action of ***Body***. (+) (Pure Precept #2)
You steal an apple.	You admit what you have done.	Go in to the next Circle.	Action of ***Speech***. (+) (Great Precept #4)
Your boss criticizes you.	You remind yourself that your boss has positive qualities.	Go to the Center.	Action of ***Mind***. (+) (Great Precept #7)

(2) ***Meditation.*** The second pillar of practice is *Meditation* (Skt. *Samahdi*), which includes Effort, Mindfulness, and Concentration (or Meditation). Mindfulness and Meditation concern the way that we attend to things in the world, including our inner (mind) states, as well as things happening around us. Right Effort is the skillful use of our bodies and minds: for example, it is sometimes helpful to relax our efforts if we find that we are working too hard to get somewhere in our training ("efforting").

The word "Zen" is Japanese for "meditation" (Skt. *djana*). Accordingly, Zen Buddhism emphasizes both formal (seated and walking) meditation and mindfulness meditation in everyday life (aka "working meditation" (MacPhillamy, 2016). In TAE, the clockwise turning of the *Dharmachakra* can be regarded as the resolution (or "cleansing") of karma through meditation — aka "turning the wheel *within*" (Jiyu-Kennett, 1999). The idea is that meditation is itself an ethical act. If done properly, it leads to purification of the mind, or development of wholesome mind states. The Buddha noted that wholesome mind states lead to wholesome speech and conduct. Thus, time spent in meditation has a cascade of benefits — it is time well spent.

In TAE, players encounter some scenarios that ask them to "Grasp the Will": if players agree, they skip three turns, rather than continuing along the path they are already traveling. Players are invited to think of these missed turns as opportunities to "sit still" (meditate). The reward is movement inwards, towards the center of the wheel (*nirvana*) and an increase in *avatar rank* (see following section for details).

(3) ***Wisdom.*** The third pillar of practice, Wisdom (Skt. *prajna*), includes the development of Right Understanding (or View) and Right Intention. Right Intention is the aspiration to understand reality as it is (not as we desire it to be) and to think and behave in ways that lead to freedom from suffering. Right View includes knowledge of the Four Noble Truths,

Eightfold Path, and three characteristics of everyday existence (*samsara*): (1) suffering, or *dukkha;* (2) impermanence or *anicca*; and (3) the interdependence of all things, or *anatta* (Jiyu-Kennett, 1999: pp. 81-3). In TAE, these teachings are implicit in the karmic narratives (Table 1) and in the "ups" and "downs" of game play, as players reflect on how meditation and preceptual practice help us to develop wise ways to respond to life events.

Within the game, the development of wisdom is represented by the progression from lower to higher ***Avatar "Ranks":*** the highest rank is that of a Buddha (fully enlightened being). The mind of a Buddha is selfless, and their actions are free from the influence of greed, anger and ignorance (the three "poisons"). Note that the game does not automatically terminate when a player reaches the highest rank: they may opt to continue playing for the benefit of others. This is a core teaching in Mahayana Buddhism: the commitment to endless training is itself the mark of enlightenment. Thus, the superficial attainment of *nirvana* (reaching the center of the wheel) is not the end goal; it is a step in the journey to help all beings become free from suffering.

After the Buddha's first lesson to his followers, he continued to expound his teachings for more than four decades (Jiyu-Kennett, 1999). The Four Noble Truths and the Eightfold Path remained the foundation of his *Dharma* (teachings) throughout this time. More than 2500 years later, they still serve as a starting point for Buddhist training and practice.

1.1.2. Conceptual and Experiential Learning in Soto Zen

TAE is an educational game: its primary purpose is to support Zen spiritual learning and practice. The gaming aspects may be supportive of this goal, but they are not the primary aim. It is therefore important to consider the types of knowledge the game is intended to foster.

In the learning sciences, researchers distinguish between two, fundamentally different types of knowledge (Ullman, 2004). The first is ***Declarative*** (aka "explicit" or "conceptual"):

that is, the type of knowledge that can be expressed in words ("declared"). In Buddhism, an example is knowing when, where, and how to meditate. Another example is the ability to talk about key concepts, such as the Four Noble Truths, the Eightfold Path, and the Buddhist Precepts. This type of knowledge can be useful as a starting point for cultivation of Wisdom. However, it does not, in itself lead to liberation from *dukkha*.

The second type of knowledge is ***Procedural*** (aka "implicit" or "embodied"): that is, skill-based knowledge that is acquired through repeated practice ("procedures"). This knowledge is implicit (unconscious) and "embodied" (e.g., "muscle memory" or the skilled habits of an expert pianist). In Buddhism, an example is the skill of doing seated meditation (e.g., *zazen* or *Shikantaza* in Soto Zen; Morgan, 2016). A second example is the ability to apply Buddhist precepts in everyday situations, automatically and without conscious deliberation. These abilities are based in procedural learning, rather than declarative (book) learning.

In TAE, procedural learning takes place through interaction with game elements that allow players to explore different ethical scenarios and to experience the consequences of their actions — e.g., the way that "good" (skillful) and "bad" (unskillful) actions lead to different consequences. Movement mechanics within the game support this kind of visceral learning: advancing along a path (clockwise, concentric movement) represents spiritual growth, whereas backward movement signifies temporary setbacks. Players thereby come to experience how different "actions" of body, speech, or mind led to pleasant or unpleasant outcomes.

If players are mindful of their inner (mind) states, they also come to experience how their attitudes of mind can lead to greater suffering (disappointment due to attachment to outcomes) or less suffering (acceptance due to letting go of attachments). The term "Enlightenment" (as in "Training and Enlightenment") can be understood as the realization of these deep, visceral truths.

1.1.3. Digitization of Zen Buddhist Training and Enlightenment

The main goal of this project was to develop TAEeG, a digital version of the TAE board game. We used Python and supporting libraries, which allow for dynamic content representation, game logic implementation, and procedural rule enforcement. In addition to representing visible features of TAE— such as the game board layout —TAEeG is designed to capture the underlying logic of the original board game, including simulated die rolls and rules that govern movement within and between tracks, changes in avatar rank, and accumulation of reward ("merit") points. We used Tkinter to build the graphical interface, which includes a digital representation of the game board and text "pop-ups," to display instructions and karmic narratives. We used a relational (SQLite) database to store the text (narratives) and rules: the database module supports dynamic retrieval and application of rules (internal game logic), and presentation of text within the graphical interface.

1.2. Structure of Thesis

The rest of this thesis is structured as follows. ***Section 2 (Background)*** reviews the history of Buddhist board games and research on "serious" games in educational technology. ***Section 3 (Methods)*** describes the design, implementation, and testing of TAEeG. ***Section 4 (Results)*** presents TAEeG (v1), focusing on software architecture, code execution, and fidelity to the software design principles and goals. It also presents a sample "run through" to illustrate what the end user would experience on executing the code. ***Section 5 (Limitations and Future Directions)*** discusses limitations of TAEeG (v1) and outlines future work. Finally, ***Section 6 (Summary and Conclusions)*** highlights key contributions of TAEeG to the fields of educational gaming and Buddhist learning and practice.

2. Background

This section summarizes relevant research in two main areas: educational technology, and the history of educational games in Buddhism and other Eastern religious traditions.

2.1. Computer-Based Educational Games

In the past few decades, there has been an explosion of computer-based learning, including sophisticated applications, such as intelligent tutoring systems (Jackson, et al., 2009; Jackson & MacNamara, 2013), which enable continuous monitoring, evaluation, and real-time feedback (Frishkoff, et al., 2016) and "scaffolding" of easy and hard problems over time and in response to learner performance (Shute & Zapata-Rivera, 2012). These applications address the need for individualized learning that is cost-effective and accessible, even in remote geographic areas.

Computer-based learning also provides new opportunities for creativity in learning and instruction. Educational (or "serious") games are of growing interest, because they stimulate play, which promotes interest and enjoyment (Jackson & MacNamara, 2013; Shute & Zapata-Rivera, 2012), alongside learning and instructional goals. This section discusses research on serious games, including the risks and benefits of "gamifying" tools for learning and instruction.

2.1.1. Serious Games in Educational Technology

Gamification is the practice of introducing elements of games, such as challenges (e.g., problem-solving, puzzles, performance monitoring) and rewards (e.g., badges, points, change in avatar rank) into non-gaming environments. As describe in the following section, serious games are particularly effective for learning material that is rote, repetitive, or highly technical ("dry").

At the same time, recent research shows that gamification of learning does not necessarily improve learning. This suggests the need for careful design of serious games.

2.1.2. Gamification of Learning: Risks and Benefits

Recent research suggests that gamified learning (aka "edugames") can support learning in a variety of domains, including STEM fields (Long & Aleven, 2014), language acquisition (Jackson & McNamara; 2013; Frishkoff, et al., 2016), healthcare (Boeker, et al., 2013; Gentry, et al., 2018), and moral reasoning, or ethics (Hodhod, et al., 2009; Hodhod, 2010; Lin, et al., 2022; McKenzie, 2013).

Several studies have used the traditional Indian board game, Snakes and Ladders, for educational applications (see Section 2.2 for discussion of Snakes and Ladders, and its precursors in ancient India). Simbolon et al. (2022) used an analog version of Snakes and Ladders to teach mathematics to 5th grade students in Indonesia. Wardhani et al. (2021) used an outdoor version of the game to promote socio-emotional development (learning and cooperation in groups) among Indonesian children, ages 7-12. Similarly, Ibam et al. (2018) developed a Java program based on Snakes and Ladders for moral education in schools.

Hodhod (2010) built the Adaptive Educational Interactive Narrative System (AINS), which teaches ethics using an immersive digital environment to present real-world ethical dilemmas. AINS is an intelligent tutoring system (ITS). It incorporates a ***Student Model*** (representation of what each learner knows, and learner characteristics that can affect the learning process), and a ***Learning (or Domain) Model*** (representation of knowledge and skills to be learned). The system uses artificial intelligence (AI) to select and present specific problems to the learner and to provide real-time feedback. This and similar work — e.g., Lin et al. (2022) and

McKenzie (2013) — demonstrate how state-of-the-art methods in educational technology can support learning in complex and ill-defined areas, such as ethics.

Benefits. The research cited above suggests that gamified learning can support learning by simulating real-world experiences in a risk-free environment, where learners can develop their skills without the risk of making mistakes with potentially serious real-world consequences. More advanced methods, such as ITS, support individualized and adaptive feedback and optimal scheduling of easy and hard examples, which have been shown to improve motivation and engagement (Jackson & MacNamara, 2013; Long & Aleven, 2014), as well as robust, long-term learning and retention (Boeker, et al., 2013; Frishkoff, et al., 2016).

Risks. Recent studies have compared computer-based learning applications with and without gamification. Results point to potential downsides of gamified learning: some features that promote increased user interest and engagement can prove confusing or distracting. For example, Jackson and MacNamara (2013) conducted a controlled experiment to determine outcomes of a traditional intelligent tutoring system (ITS) with and without gamification. Outcome measures included motivation and engagement metrics, as well as learning (quality of self-explanation in a language comprehension task). On the whole, gamified and non-gamified systems led to similar task performance (learning). In addition, the game-based system led to higher levels of self-reported enjoyment and motivation. However, there was a caveat: earlier in the task, the gamified system led to decreases in learning, when compared with the traditional ITS. The authors speculate that some gamification features may be distracting to learners, particularly at early stages of learning.

Dankbaar et al. (2016) studied the use of computer simulations for teaching emergency medical team (EMT) skills, compared with “business as usual” (actual hands-on training). They

found that computer-based simulations were as effective as hands-on experience. However, the use of high-fidelity (more detailed) simulations resulted in worse outcomes, when compared with low-fidelity (less realistic) simulations. They speculated that the high-fidelity simulations included more distracting features, which resulted in divided attention and lower learning and retention. This result adds to findings from Jackson & MacNamara (2013), suggesting that complex graphics can detract from learning applications.

These findings illustrate that educational games must carefully balance entertainment elements with educational effectiveness. While engaging mechanics, such as rewards, graphics, and narrative immersion, can improve student engagement, they may also introduce potential distractions that detract from learning outcomes. Likewise, excessive focus on high-paced action, overly complex rules, or gamified motivations that prioritize outside rewards over intrinsic learning can shift players' focus away from educational content (Carvahlo, et al., 2013).

In summary, recent research shows that gamified learning can have costs, as well as benefits. This underscores the need for more rigorous studies, including careful comparison of learning with and without gamification (i.e., the use of control groups), and model-based frameworks for design and implementation of serious games (Carvahlo, et al., 2013)

2.2. Educational Buddhist Board Games

This section describes the broader context for TAE as a serious game, beginning with a brief history of analog (board) games in Buddhism and other Eastern religious traditions. It then describes the TAE board game in detail. To our knowledge, TAE is one of only two Buddhist board games that are available in English and are primarily designed for education, rather than entertainment (the other board game is BuddhaWheel™; Rogers, 2004).

2.2.1. A Brief History of Eastern Religious Board Games

Board games have a long history in Buddhism and other Eastern religious traditions. This section briefly summarizes this history to give a broader context for understanding the content and structure of TAE.

The TAE Board Game was conceived and designed by Reverend Master Jiyu, a Zen Master who was ordained and trained at Sojiji Monastery in Japan and later founded the Order of Buddhist Contemplatives (OBC). Notes from her diary suggest that she came into contact with Buddhist board games, similar to TAE, when she was training in the East (see Box 1 insert). This points to the connection between modern Eastern religious board games and the history of these games, which date back to at least the 12th century (and, as mentioned previously, imagery such as the Dharmachakra is even older). The following section briefly reviews this history, focusing on precursors of the modern "Snakes and Ladders" (or "Chutes and Ladders") game, which remains an extremely popular game for introducing young children to basic ideas of "good" and "bad" action (moral virtue, or ethics), and the Bhavachakra, an image depicting core Buddhist principles, which was reportedly developed by Shakyamuni Buddha as an early teaching tool.

2.2.1.1. Snakes and Ladders

Snakes and Ladders (also known as "Chutes and Ladders") is a popular children's game that was first made available in English and distributed by Milton Bradley in 1943. The game board consists of a grid; each square represents a life event, and the squares are numbered from lowest to highest, starting in the bottom left corner and culminating in the upper left corner of the board. Player markers (avatars) begin the game in the first (lower left) square and roll a die to determine how many squares to move on each turn. The goal is to be the first one to reach the final (upper left) square. When players land on a square representing a virtuous action (e.g., they

give their brother a hug), they are rewarded by moving up a ***Ladder*** to a higher square. When they land on a square representing a selfish or mischievous deed (e.g., they steal a cookie), they are shuttled back in the opposite direction (***Snakes*** or ***Chutes***, depending on the edition). The game teaches basic lessons in morality (good and bad actions, and their real-life consequences).

2.2.1.2. Gyãn Caupar

Snakes and Ladders is based on Gyān Caupaṛ, a game that originated in Western India in the late 17th or early 18th century (Bado-Fralick & Norris, 2010; McGuire, 2014, 2022; Schlieter, 2012; Schmidt-Madsen, 2019). Schmidt-Madsen (2019) suggests that the formal structure of Gyān Caupaṛ (sequentially numbered grids with snakes and ladders) was influenced by other games, including the 15th century Italian *gioco dell'oca* ("Game of the Goose") and the 12th century Chinese *xuanfo tu* ("Table of Buddha Selection").

2.2.1.3. Xuanfo Tu

According to McGuire (2022), the Chinese board game *xuanfo tu* was designed as an alternative to secular games that had a similar structure, but were focused on wordly attainment (e.g., gaining positions of power within the Chinese bureaucracy), rather than spiritual growth. Like TAE, *xuanfo tu* emphasizes the three "pillars of practice" in spiritual development: Meditation, Ethics, and the cultivation of Wisdom (McGuire, 2014; McGuire, 2022; Ngai, 2011).

2.2.1.4. Bhavachakra ("Wheel of Life")

Another image that is commonly used in Eastern religious training is that of the *Bhavachakra* ("Wheel of Life" or "Wheel of Becoming"). Shakyamuni Buddha is said to have developed the Bhavachakra to convey subtler teachings in Buddhism, including the 6 Cosmic Realms (Jiyu-Kennett, 1999) and a detailed explanation of Karma, known as the Cycle of Dependent

Origination (MacPhillamy, 2016). According to legend, the Buddha requested that that the image be painted according to his specific instructions (Khantipalo, 2013). He further asked that the image be displayed in every Buddhist temple, so the teachings would be easily accessible to everyone, even those who were illiterate (which would have included most people at that time).

The BuddhaWheel™ board game (Rogers, 2004) is based on the Bhavachakra (Fig. 2). Like TAE, it was designed for educational purposes and is a cooperative game.

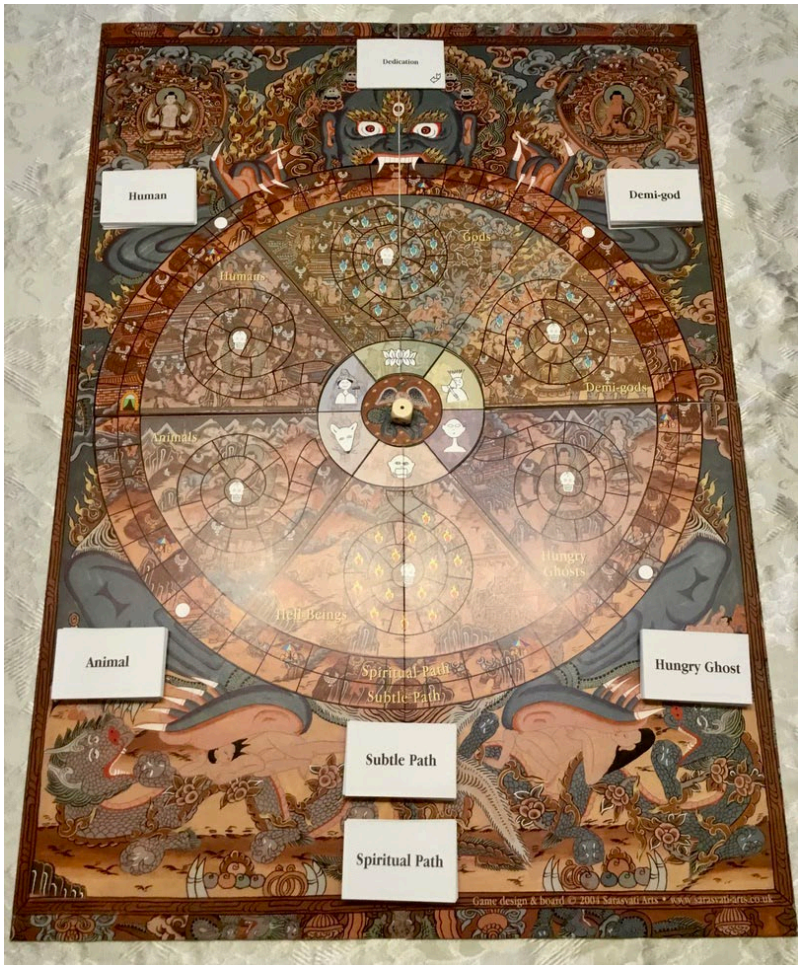

Figure 2. BuddhaWheel© Game Board.
The BuddhaWheel (Rogers, 2004) game board depicts the Bhavachakra (Wheel of Existence). The three higher realms (Human, Heaven, and Asura Realms) make up the top half of the wheel. The three lower realms (Hungry Ghost, Hell, and Animal Realms) make up the lower half. The three poisons (Snake = aversion, Bird = attachment, and Pig = ignorance) are depicted at the center of the wheel.

2.2.2. Training and Enlightenment (TAE) Board Game

Training and Enlightenment (TAE) introduces players to core principles and practices in Soto Zen Buddhism (Jiyu-Kennett, 1989). As detailed in this section, TAE incorporates features of the Dharmachakra (Wheel of Training): this is the image that represents the main Path of Training. The spokes of the wheel represent the Noble Eightfold Path (Figure 1). It also incorporates elements of the Bhavachakra (Fig. 2), including the Six Cosmic Realms (Heaven, Hell, etc.) and the Three Poisons (Greed, Anger, and Ignorance). The concept of "ladders" (accelerated progress along the Path) and "chutes" (regress) is also implicit in the game mechanics. Specific elements of the game board, and their significance, are detailed below.

The TAE game includes several forms of documentation, including an Instruction Manual and a Quick Reference Guide. See Appendix A (Table 9) for a semi-structured table with individual Instructions, coded to align with different components of game play. The package also includes a separate sheet with a list of Buddhist Precepts, which (as indicated at the top of the sheet) can be understood as interreligious vows — that is, intentions to cultivate virtues, such as compassion, love, and wisdom (*Three Pure Precepts*) and virtuous qualities, such as honesty, humility, sobriety, and so forth (*Ten Great Precepts*). See Appendix A (Table 8) for a complete list of Precepts. Finally, there is a record-keeping sheet, with a fillable table for tracking players and changes in player variables, such as merit points and player marker (rank). The record-keeping sheet can be used to continue game play over multiple sessions.

2.2.2.1. Physical Game Components

In addition to documentation, the TAE game includes five physical game components:

1. **Avatar Markers**
2. **Die**
3. **Game Board**
4. **Karma and Bodhisattva Cards**
5. **Reward cards (Merit Points)**

1. Avatar Markers. At the beginning of each game session, each player chooses one of three Avatar Markers: a pig, a rooster, or a snake. In Western practice, the pig represents greed (***attachment***), the snake represents anger (***aversion***), and the rooster represents ignorance (***delusion***).[1] As shown in Figure 3, each player marker belongs to one of six Avatar Ranks (Jiyu-Kennett, 1989):

- Rank 1: worldly minded beast (upside-down pig, snake, or rooster)
- Rank 2: spiritually minded beast (right-side-up pig, snake, or rooster
- Ranks 3– 5: Bodhisattva (awakening being, who vows to help all beings to awaken):
    - Avalokiteshwara (Bodhisattva of Compassion)
    - Samantabhadra (Bodhisattva of Unconditional Love)
    - Manjusri (Bodhisattva of Wisdom)
- Rank 6: Maitreya Buddha (Fully Awakened Being; aka "The Buddha to Come")

The six Avatar Ranks can be viewed as ***attitudes of mind*** that can be brought to practice (cf. Jiyu-Kennett, 1999: Chapters 5-7). The goal of practice is to move from a worldly-minded attitude, which tends to increase suffering, towards the mind of a Bodhisattva — the expression

[1] Note: In traditional Indian religious teachings, the pig is associated with delusion, and the bird with attachment.

of compassion, love, wisdom, and selflessness. Eventually, in Mahayana Buddhism, the goal is to develop the qualities of a Buddha (represented here by Maitreya, the "Buddha to Come"): this represents fully enlightened (i.e., selfless and wise) thought and action.

Figure 3. TAE Avatar Markers (Ranks).
Avatar markers represent different "attitudes of mind" in Buddhist training (Jiyu-Kennett, 1989).

2. Die. Players roll the die to initiate each Turn and advance the number of spaces indicated by the die roll. Die values range from '0' to '8'. A die roll of '0' indicates that the player should "sit still" (i.e., meditate) for that turn.

3. Game Board. As shown in Figure 4, the original TAE game board features a central image, which is that of a wheel (Skt. -*chakra*). The wheel comprises three concentric circles ("Paths") and a central circle ("Hub"). Each circle (path) is divided into "spaces," representing positions along the Path of Training. Some spaces along the Path are blank, and some spaces have printed text that describes a simple, everyday scenario (e.g., "you steal an apple and then

decide to return it") and the consequences (game-based actions) associated with the scenario (e.g., "move forward 3 spaces" or "go in one Path"). See Table 1 for additional examples.

Figure 4. TAE Game Board.
Central image depicts the Dharmachakra (Wheel of Training),
with spokes representing the *Noble Eightfold Path* (cf. Fig. 1).
The *Six Cosmic Realms* surround the Wheel of Training (cf. Fig. 2).

4. Karma and Bodhisattva Cards. Some spaces along the Path instruct players to choose from a shuffled stack of either "Karma" (orange-colored) or "Bodhisattva" (green-colored) Cards (see Figure 1, bottom left corner). There are 30 unique Karma Cards and 38 Bodhisattva Cards. See Table 1 for examples of card contents.

5. Reward Cards (Merit Points). The TAE package includes a deck of gold cards, representing different *merit point* values (1, 7, or 56 merit points). In TAE, these points are

awarded for certain meritorious actions, representing an increase in *positive karma* (= *merit*). In TAE there are no corresponding cards or tokens representing negative karma.[2] However, some Karmic Actions result in a loss of merit points. Negative karma may also manifest as the loss of avatar rank, or movement backwards — away from the center of the wheel, or back out to one of the six cosmic realms (i.e., outside the Wheel of Training). The specific rules that determine karmic consequences in TAE are described in the following section.

2.2.2.2. Game Play Logic

The TAE game logic includes the following sets of rules:

(A) Rules to Start the Game (Avatar Marker, Starting Spoke)

(B) Movement along the Path (Wheel of Training)

(C) Karmic Narratives (*Karma–Vipaka* sequences; Karma/Bodhisattva Cards)

(D) Movement between Path layers (*Ladders* and *Chutes*)

(E) Movement outside the Path (Cosmic Realms)

(F) Rules for Changing Avatar Rank

(G) Rules for Gaining, Losing, and Transferring Merit Points

(H) Rules for Meditation ("Lost" Turns)

(I) Rules to Exit or End the Game

(A) Rules to Start the Game. To start the game, a player selects a card with an image of a pig, snake, or rooster (wordly minded avatar marker). The marker is placed upright in a wooden stand, so the avatar marker can be moved along the game board. The starting location is the

[2] Two contemporary games — BuddhaWheel™ (2004) and Cosmic Karma™ (2011) — have game mechanics that represent the accumulation (and "burning off") of negative, as well as positive, karma.

Human Realm (Figure 4: White Circle in bottom right-hand corner of the game board). To enter the Path of Training, the player rolls the die to determine how many spaces to move. To enter the Path, they place their marker at the end of one of the eight spokes (areas of practice): Action, Speech, Livelihood, Mindfulness, Meditation, View, or Intention (cf. Figure 1).

(B) Rules for Movement along the Path. Once players enter the Path (Wheel), they roll the die to initiate each turn. A die roll higher than '0' indicates how many spaces to move forward. The default direction for movement is clockwise and concentric: that is, players begin by moving their avatar along the outer path from left to right, until they are prompted to move inwards to the next (intermediate) path. Once they enter the second path, they continue moving left-to-right along that path, until they are prompted to move inwards to the third and final (inner) path. The ***outer path*** contains 86 spaces; the ***intermediate path*** contains 63 spaces; the ***inner path*** contains 47 spaces. The ***center (hub)*** counts as one additional space and is connected to the three surrounding paths by ***eight spokes***, or connected sets of spaces, which allow for movement between paths. Thus, there is a total of 205 spaces (8 outer Spokes + 196 Spaces along the Path + 1 Center space).

(C) Karmic Narratives. As players move through the game, they encounter karmic "narratives" (action-event sequences; cf. Table 1). Narratives are printed on non-blank spaces along the Path of Training, and on Karma (and Bodhisattva) Cards, which players are instructed to draw when they land on certain spaces.

Some narratives result in undesirable consequences, such as loss of rank, lost turns, or lost points. English-speakers casually refer to these outcomes as "negative karma" (or simply as "karma"). Technically, this usage is incorrect: the Sanskrit word *karma* means action (not the

effect of an action). In Buddhism, *karma* is specifically understood to reflect an intentional action of body, speech, or mind (Jiyu-Kennett, 1999). The consequences of such an action (Skt. *vipaka*) depend on the underlying intention: according to Buddhism, an action undertaken with wholesome intent — to benefit oneself or others, or both — has positive, or "meritorious" karmic consequences (NOTE: in real life, these consequences may be immediately apparent, or they make take time to bear fruit[3]). In TAE, game-based rewards are represented by a gain in merit points, forward movement along path, or an increase in avatar rank. Game-based penalties are represented by a loss of merit, backward movement, or a decrease in avatar rank.

(D) Movement between Path layers (*Ladders* and *Chutes*). As players move along the Path, they may land on a space that is connected to one of the eight spokes. They are then instructed to move in one path (e.g., from the outer to the intermediate path or from the inner path to the center). In this way, spokes function as ***Ladders***. In addition, some non-spoke spaces have instructions to move in one path. Other spaces trigger backwards movement (away from the center hub) — either counterclockwise or "back to the beginning" (out to one of the eight spokes or one of the Six Realms). We use the term ***Chute*** to refer to these backward path connections (see Appendix A: "Path Connections Table" for list of Ladders and Chutes.)

(E) Movement outside the Path (Six Cosmic Realms). The four corners of the game board depict the Six Realms of Buddhism, which can be interpreted as psychological states that are more or less "karmically loaded"[4] (Jiyu-Kennett, 1999). At the start of the game, before

[3] Delayed karmic consequences could be supported in future computer-based versions of TAEeG, leading to more realistic simulations of real-world karma.

[4] As illustrated in Figure 2, the Human, Heavenly, and Titan Asura realms are considered the "higher" (more desirable) realms, and tend to be associated with lighter (less negative) past karma. By contrast, the Animal, Hell, and Hungry Ghost Realms, are considered "lower" (less desirable). The implication is that beings who have accumulated a large amount of (negative) karma are more likely to be born in one of the

entering the Wheel, players place their avatar markers in the Human Realm. Once they enter the Wheel, they continue moving in a clockwise, concentric direction as described above, unless they are prompted to return to the Six Realms. When a player is sent out to one of the Six Realms, they must move through each of the realms before returning to the Wheel.

(F) Rules for Changing Avatar Rank. Recall that the six avatar "ranks" represent different attitudes of mind: *Rank 1,* ***wordly mind*** (upside-down rooster, pig, or snake); *Rank 2*, ***spiritual mind*** (right-side up rooster, pig, or snake), *Ranks 3-5*, ***Bodhisattva attitude of mind*** (compassion, love, or wisdom); and *Rank 6*, ***fully awakened mind*** (Buddha mind). There are three events in TAE that trigger a change in avatar rank. First, if a player lands on a "Grasp the Will" space and decides to sit still (meditate) for three turns, they are rewarded with an increase in rank. Second, some (Karma or Bodhisattva) cards reward positive actions with an increase in rank. Finally, if a player reaches the center (hub) and decides to return to the start and keep playing, they increase their rank.

(G) Rules for Gaining, Losing, or Transferring Merit Points. Players receive ***merit*** points (*positive karma*) when they land on a space or draw a card that represents a wholesome (ethical) action. Once player avatars achieve the rank of a Bodhisattva or of the Buddha to Come, they can use merit points to offset negative karmic consequences. They may also "transfer" their merit to other players: that is, a player that has achieved the rank of Avalokiteshvara (or a higher rank) can transfer merit, in the spirit of compassion and cooperative game play (cf. story in Box 1).

---

lower realms. The Human Realm is regarded as the most auspicious realm to be born into although there is *dukkha* (suffering), as there is in all 6 realms, there is just the right amount: not so much as to make it impossible to engage in spiritual practice and not so much that there is no incentive to practice.

(H) Rules for Meditation ("Lost" Turns). When a player lands on spaces with instructions to "Grasp the Will," they are offered two choices: (A) they may choose to continue along the same Path (1st, 2nd, or 3rd circle) on the next turn; or (B) they may wait three turns and then advance to the next innermost circle. Choosing (B) corresponds to what Jiyu-Kennett (1989) referred to as "Grasping-the-Will": in electing to "miss" three turns, players are choosing to "sit still" (i.e., meditate), rather than continuing along the Path in the same manner as before. The reward for sitting still is accelerated movement along the path (inward/upward to the next "level"). Some spaces instruct a player to lose one or more turns. The "loss" of a turn can also be regarded as an opportunity to sit still (i.e., meditate), rather than a penalty.

(I) Rules to Exit or End the Game. When players reach the center (hub) of the Wheel, they are asked to make a decision: they may either remain in the center for the duration of the game (taking themselves out of subsequent play), or they may return to the Wheel of Training and continue game play. The decision to "opt out" of further play is symbolic of the choice to withdraw from the world of *samsara* (ordinary existence). The decision to continue game play symbolizes the path of the Mahayana Buddhist, who commits to staying in the world and continuing to train until all beings everywhere are free from suffering. In Zen Buddhism, this is what is meant by the teaching that "Training and Enlightenment are One" (Dogen's *Shushogi*): according to this idea, enlightenment is viewed as a process, rather than as a static state, and choosing to train for the benefit of all beings (selfless action) is the very heart of enlightenment.

BOX 1: Development and Manufacturing of the Training & Enlightenment Board Game™

Reverend Hugh Gould is a Zen Master in the Soto Zen tradition. He was ordained as a Buddhist Monk in 1988 by Reverend Master Jiyu-Kennett, who founded the Order of Buddhist Contemplatives (OBC). Rev. Hugh received the Dharma Transmission in 1997 from Rev. Master Daizui MacPhillamy, former head of the OBC and one of Rev. Master Jiyu's senior disciples. Rev. Hugh was named a Dharma Teacher in 2000 and a Zen Master in 2003. In June 2021 he was appointed as Prior (Head Monk) of the Eugene Buddhist Priory in Eugene, Oregon.

Reverend Master Hugh Gould

The following interview was conducted by Nithi Deivanayagam at the Eugene Buddhist Priory on March 2, 2024. We thank Reverend Master Hugh for his time and gracious commentary on Soto Zen practice and the development, manufacturing, and real-world use of Master Jiyu's Training and Enlightenment board game.™

ND: How did Reverend Master Jiyu-Kennett come up with the idea for an educational Zen Buddhist board game?

RH: In 1962, she left England to go become a Monk. And the first thing she did was go to Malaysia, where she trained in Chinese Buddhism. My understanding is that while she was in Malaysia, she encountered similar types of board games. Later she trained at Sojiji Monastery in Japan— but I don't remember her saying that she saw any board games in Japan. So, I think it was Malaysia where she first had the idea for an educational Buddhist board game. I'm guessing

she probably studied the games or just had a feeling for how they worked. And while she was there, in the Far East, she kept a journal and wrote all sorts of stuff down that she had seen.

ND: When and how did Rev. Master Jiyu introduce this idea to the West?

RH: In 1969 she moved to Northern California and started one of our two main monasteries, Shasta Abbey (the other Monastery, Throssel Hole Abbey, is located in the U.K.). The property that she bought was located at the base of Mount Shasta, in a beautiful, semi-rural area near the base of the mountain. Originally, dating to the 1920s and 30s, it was used as a motor motel, and they had these little stone cabins around the property that people would rent. When our Monks built the monastery, they constructed walkways between the existing buildings, to create a cloister. In one of the buildings, there were wooden floors, and Master Jiyu had a picture of the Training & Enlightenment game board painted directly on the floor. So, the game was originally conceived and played on a wooden floor in this one building. That's where it started. I wasn't there at the time, so this is my understanding of what happened in the past, based on stories I heard when I moved to the Abbey in the 1980s.

ND: How did you contribute to the development and manufacturing of the board game?

RH: As a junior Monk, I worked in what we called the Buddhist Supplies Shop. We had a fairly large, active mail order business, where we'd sell Buddhist books, meditation supplies, incense, gongs, statues, and so forth. And at one point while I was working there, Reverend Master Jiyu decided she wanted to create and distribute a board game. And the senior Monk who was in charge of the shop was back and forth with Reverend Master Jiyu about how they wanted to do it. He would find different businesses in town to produce the various things that they wanted. We had one company that built our meditation benches, and we had them make the little

wooden holders for the avatars. And then there was a local printer who printed the game board for us, and the various avatars and stuff. So, it was pretty nuts-and-bolts — very practical stuff.

ND: Gwen mentioned that there are around 300 existing copies of the Training & Enlightenment Board Game.™ How are they currently used in your Soto Zen practice?

RH: We've played the game at different priories in the past. For example, there was one Buddhist Prior back in England who loved playing the game. And once a month, or so, he'd have a game day: the congregation could come, and he would be there, and they'd all play the Training & Enlightenment game™ together. And in the course of game play, people would ask him questions, and he would address the spiritual underpinning of what was going on in the game. And the other thing too that comes to mind is that back again at the monastery in England, traditionally there's *Wesak*, the Festival of the Buddha's Birth, which we usually celebrate in the springtime, often on a Sunday in May. So, on the Saturday before *Wesak*, we would have a family weekend. And any congregation members who had children could bring their whole families and come for the weekend. And on that Saturday evening, I would have the Training and Enlightenment ™ game available. And I would play it with the kids who were interested.[5]

ND: Can you talk about a particular time when you played the game, and a Soto Zen teaching that was revealed during game play?

RH: There's a point in the game where you earn *merit points* [positive karma]. And when you get to a certain avatar rank, that of *Avalokiteshvara* [the Bodhisattva of Compassion], you're allowed to offer merit points to other people, to offset negative karma when they land on certain spaces or draw certain cards. So, during one game, I ended up getting to that level and I had merit points. And then some kids started landing on spots that were associated with bad karma.

[5] "News of the Order: Throssel Hole Buddhist Abbey." In *The Journal of the Order of Buddhist Contemplatives*, *33*(2), 2560 B.E. (Summer 2017), ISSN 0891-1177, p. 51.

And I'll never forget that when I offered merit to this one young girl; she was just so moved that somebody was offering this help to her. So, in one respect, this game is actually about how we interact with people, how we interact with the world around us. And one lesson, one thing that you find out in the game, is that acting selfishly just doesn't go anywhere.

ND: Do you have to be Buddhist to understand or get something out of playing it?

RH: So, these kids I played the game with — most of them had no knowledge of Buddhism. It doesn't matter. Just sit down and play the game. And by playing the game, you start learning about the nature of Buddhist practice. To me, the game stands on its own. It's sufficient in itself. Maybe some of the kids or some of the people who play the board game — they might think: "That's really interesting. I'm going to look more into Buddhism, or a related spiritual practice."

ND: What do you think about the utility of an e-version of the game, or its limitations?

RH: The game teaches basic Soto Zen ideas and practices, and it encourages you to be generous and helpful. And you'll still get that playing the e-version of the game. But it's really cool sitting around a physical game board with a group of people. When we were first designing and testing the game, we had an opportunity to try it out with Reverend Master Jiyu herself. And she was an amazing teacher — an amazing human being. And so here I was playing this game, and I'd land on different spaces on the board, and I'd ask her: "Well, what does that mean?" So here I had human teacher, a living Zen Master, that I was able to direct these questions to. It would be interesting to think about how to replicate that type of experience when people are playing the digital game. If we do end up with an app [e-version of the game], maybe we could have a hotline. And when a question comes up, users could phone a Monk and ask them questions about the spiritual meaning of a particular aspect of game play.

3. Methods

This section discusses the development and implementation of the *Training and Enlightenment e-Game* (TAEeG, version 1.0). ***Section 3.1*** describes the design and conceptualization of TAEeG and software design principles and specifications for version 1. ***Section 3.2*** describes project implementation. ***Section 3.3*** describes code testing and debugging, including unit testing, simulations, and iterative debugging. ***Section 3.4*** describes the final code execution, code storage, and versioning of software.

3.1. Design and Conceptualization

This section provides an overview of TAEeG software design (3.1.1), describes the digital game contents (source data; 3.1.2) and structure (code modules; 3.1.3), and outlines goals for implementation of TAEeG version 1 (3.1.4).

3.1.1. Overview of Software Design

Figure 5 shows the TAEeG software design, which includes three main components:

1. **Digital Game Contents (Source Data)** — cf. Appendix A
2. **Digital Game Structure (Python Code)** — cf. Appendix B
3. **Graphical User Interface (GUI) Display** — cf. Section 4 (Results)

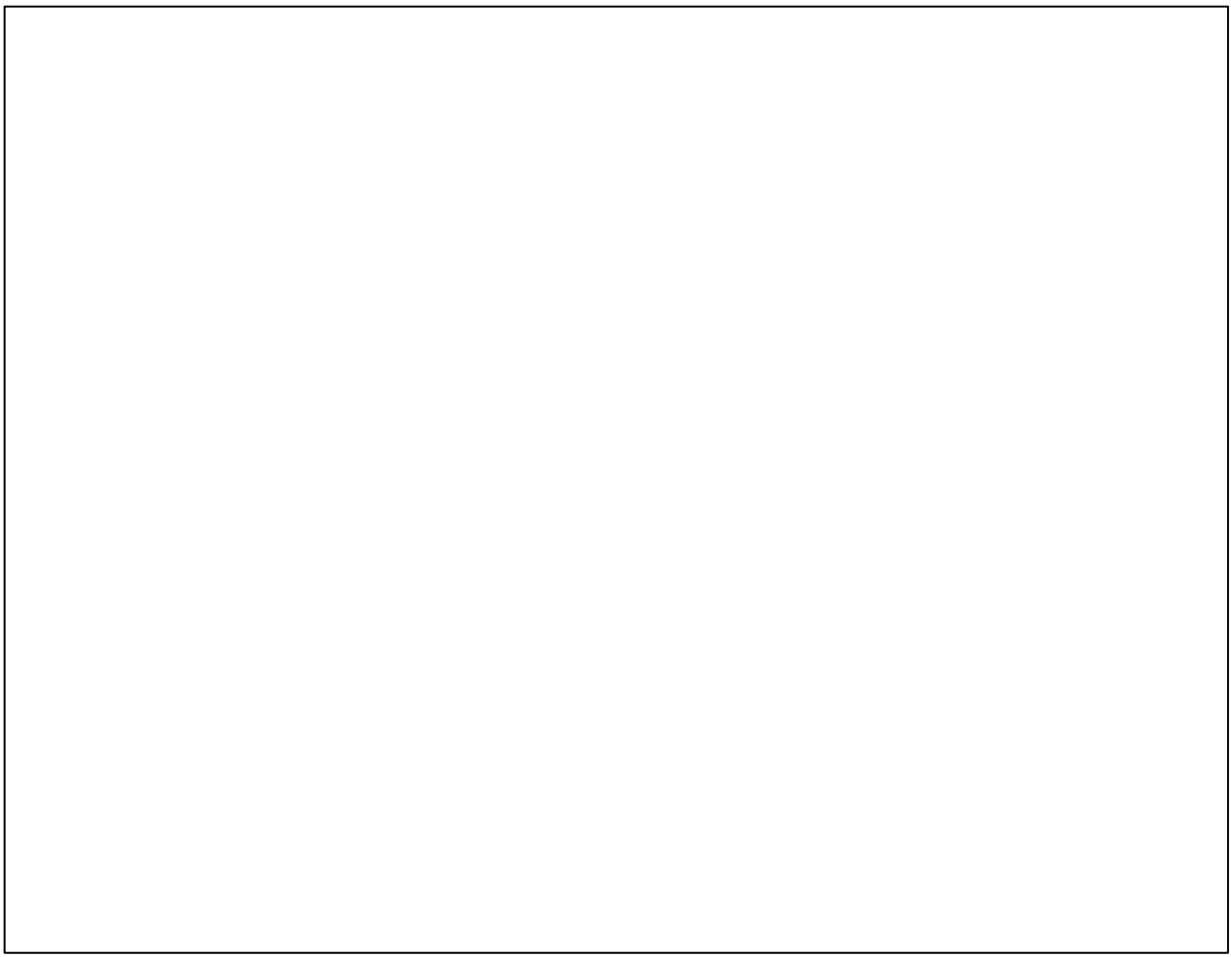

Figure 5. TAEeG Software Architecture.
(**1**) "Domain Expert"[6] performed knowledge and data capture, resulting in source data file (**Appendix A**).
(**2**) Programmers developed and implemented Python code (**Appendix B**).
(**3**) End User (Player) interacts with GUI using a desktop computer and display monitor.

Digital Game Contents (Source Data). Contents of the original TAE game are fully captured in a source data file, which includes semi-structured (tabular) data. See Appendix A for details. The source data file was created by the "Domain Expert"[4] for this project (GF) and added to a GitHub repository, together with a GitHub tasks (or "Issues") list and documentation (Technical Reports describing the analog game structure and contents).

[6] The term "domain expert" is used in a technical sense: the person who served this function (GF) was familiar with the TAE game and had the skills to capture the game structure and content as semi-structured data. These data were then used by programmers to create the TAEeG software.

Digital Game Structure (Code Modules). Two programmers (ND and a student collaborator) used selected source data to design and implement the TAEeG software. The code has a modular structure, as described in ***Section 3.1.3*** (cf. Appendix B). The GUI is coded using Tkinter, a Python library. Version 1 (TAEeG_v1) supports a single player, who interacts with the game interface using a desktop computer and supporting software.

Graphical User Interface (GUI) Display. TAEeG (version 1) is designed for use on a desktop computer, with a monitor that is sufficiently large to display the virtual game board (approximately 8" or larger). Contact authors for step-by-step instructions to end users on how to download, install, and run the software.

3.1.2. TAEeG (v1) Software Specifications

Version 1 of TAEeG (TAEeG_v1) is a partial implementation of the original TAE game. The software specifications for v1 differ from the original board game, as follows:

1. **Single Player**. TAEeG_v1 supports a single player. As described in Section 5, future versions will support multiple players, and interactions between players.
2. **Limited Storage and Tracking of Data**. In the original game, players use a Game Record Sheet, which is provided with the game board and other game pieces, to track changes in the following variables at the beginning and end of each session:
    a. ***Avatar marker*** (worldly minded, spiritually minded, etc., representing changes in *attitude of mind* as avatar progresses along the spiritual path)
    b. ***Number of merit points*** (representing the accumulation of positive karma)
    c. ***Number of "lost turns"*** (representing opportunities to *meditate*)

    In TAEeG version 1, there is no tracking and storage of these data across turns. Dynamic updating and storage of these data will be added to future versions of the software.

3. **Fixed Avatar Marker**. There is no dynamic tracking and storage of this variable. Each player avatar has a single (fixed) value in v1. Avatar ranks will be added to v1.5.
4. **No Merit Points**. Merit points are not tracked and stored in v1. Thus, there are no active rules for use of merit points to ameliorate undesirable outcomes (e.g., lost turns or being sent back to the beginning). Likewise, there is no *transfer of merit* (using points to help another player, as described in Box 1). Merit points will be added to v1.5.
5. **Cosmic Realms Inactive**. In the original TAE board game, players are initially located in the Human Realm; on their first die roll, they select one of the eight Path Branches to enter the Wheel of Training. During subsequent game play, they may be sent back out to one of the 6 Cosmic Realms (see Section 2.2.2.2, subsection E for details). In TAEeG (v1), the Cosmic Realms are displayed in the GUI, but are not used in game play.
6. **Subset of Karmic Narratives**. Version 1.0 (v1) only presents karmic narratives associated with TAE Karma and Bodhisattva cards. Narratives linked to game board spaces are not implemented. In addition, some narratives were excluded from v1 because they are complex and harder to implement. For example, some scenarios affect multiple players (which does not apply in v1), and some have multiple contingent outcomes (e.g., the player might be asked if they are "eager to get to the center"; the karmic consequence is different for "yes" and "no" responses). The full set of karmic narratives ('space' and 'card' contents) will be added to Version 2.0.

3.1.3. TAEeG (v1) Software Design Principles

In this project, we aimed to respect the following software design principles:

1. Fidelity to the original TAE game mechanics
2. An intuitive and accessible graphical user interface (GUI)

Fidelity to the original TAE game mechanics. A core principle was fidelity to the original, analog version of TAE. This implied that fundamental mechanics—such as moving through realms, encountering karma and bodhisattva events, and progressing toward enlightenment — should be carefully preserved. For example, we aimed to recreate the cyclical and non-linear movement patterns supported by the physical game. We also aimed to implement the rules governing die rolls, avatar movement, and karma/bodhisattva card effects to recreate the original game logic, as far as possible.

An intuitive and accessible graphical user interface (GUI). We also aimed to create an intuitive and accessible GUI, including a digital recreation of the game board. We aimed to design and implement virtual "buttons" that players could click on to initiate core actions, such as rolling the die and accessing the pause menu. We also explored the use of pop-up message boxes, to provide real-time feedback on avatar movement and to display text, including instructions to players and the contents of karmic action-effect (narrative) sequences.

3.2. Implementation

This section details the implementation of TAEeG (v1), including software resources (3.2.1), coding of each module (3.2.2), turn management (3.2.3), and error handling (3.2.4). Testing and debugging are described in the following section (3.3).

3.2.1. Software Resources

For this project, we used the following software and resources:

- **GitHub**: The GitHub Issues tracker was used for project and task management, and the GitHub repository (/tae_egame) was used for storage and versioning of project resources, including:
    - Software code (/tae_egame/code)
    - Source data (/tae_egame/database)
    - Documentation (/tae_egame/documentation)
- **Python 3**: Python was used for TAEeG code development, together with the following libraries and methods:
    - **Pandas**
    - **SQLite3**
    - **Tkinter**
- **PyCharm**: PyCharm was used as a code development platform, and PyCharm debugging was used to identify and resolve errors in the code.

Python is a popular object-oriented language that supports modular software design. Game elements, such as the *avatar*, *board*, *rule*, and *card*, are represented as class structures, which makes the codebase easy to manage and debug. The modular design also supports future changes, while maintaining the integrity and stability of the core architecture. The ***Pandas*** library was used for structured import, transformation, and insertion of data into an SQLite database. ***SQLite3*** was used for database storage and management (Section 3.2.2.4). ***Tkinter*** was used, with the **Canvas** widget, to create the GUI (Section 3.2.2.5).

3.2.2. Code Modules

As shown in Figure 4, TAEeG includes five code modules: A **Player Module**, an **Avatar Module,** a **Game Module**, a **Database Module**, and a **GUI Module**.

3.2.2.1. Player Module

The **Player Module** defines how users interact with the game environment, including movement, interactions with cards, and engagement with game challenges through card content (cf. Figure 5, Module 1).

3.2.2.2. Avatar Module

In version 1, the **Avatar Module** manages player avatar attributes, such as type and rank, and position on the game board. The **Avatar** class (object) initializes and stores key attributes, including starting position and radius, and manages avatar movements and interactions with other game objects. As shown in Figure 5, changes in Avatar Movement are tracked within the Game Module (see Section 3.2.2 for details) and displayed in the GUI (Section 3.2.4).

3.2.2.3. Game Module

The **Game Module** implements the core logic of TAEeG and is closely connected with the other three modules. Callback functions are used to control interactions between the Database (DB), GUI, and Game Modules, as illustrated in Figure 5.

The Game Module includes an ***Avatar Movement*** submodule, which determines how avatars change position within the game environment. As illustrated in Figure 6, the avatar's position is updated based on die rolls and rules associated with specific karmic narratives, and these changes trigger updates the to the GUI to reflect changes in avatar location and presentation of new text (e.g., new Instructions or Karmic Narratives).

Figure 6. TAEeG Game Module (Overview).
Schematic representation of interactions between the Avatar Movement and Rules submodules of the main (Game) module. Inset picture illustrates how GUI would display avatar movement triggered by Rule 001: if avatar lands on one of the spokes, they move along the spoke to the next innermost path.

A list of ***Rules*** that represent game mechanics implicit in the original TAE game are stored also within the Game Module. Rules can be divided into the following categories, based on how they affect game play mechanics (cf. Appendix A: Table 4). Rules that are inactive in TAEeG (v1) are marked with an asterisk (*):

- TAE Rules that Offer Opportunities to "***Meditate***" (Lose Turns)
    - (*) Die_roll = 0: "Sit still" (i.e., meditate) for current turn
    - Rule 008: "Lose" (XX number of) turns
    - (*) Rule 002: "Grasp the will" (skip 3 turns)
- TAE Rules that Trigger Forward Movement (***Ladders***)
    - Rule 001: (landed on spoke) Go in one path (traveling along spoke); references *path_connections* look-up table (Appendix A: Table 2)
    - Rule 003: For forward (XX number of) spaces
    - Rule 006: Go in one path (e.g., outer → intermediate); references *path_connections* look-up table (Appendix A: Table 2)
    - Rule 016: Go straight to Center (glimpse of *nirvana*, or "Kensho")
- TAE Rules that Trigger Backward Movement (***Chutes/Snakes***)
    - Rule 004: For backward (XX number of) spaces
    - (*) Rule 007: Go out one path (e.g., inner → intermediate); references *path_connections* look-up table (Appendix A: Table 2)
    - (*) Rule 010: Go out to (one of the 6) cosmic realms
    - (*) Rule 011: Go to end of (one of the 8) spokes (e.g., beginning of Right Speech)
- TAE Rules that Trigger a ***Card*** Draw
    - Rule 005a: Select a (random) Karma Card (call to 'cards' DB)
    - Rule 005b: Select a (random) Bodhisattva Card (call to 'cards' DB)

Figure 7. TAEeG Game Module (Source Data: Rules).
Semi-structured list of rules and pseudo-coded functions (Appendix A: Table 6) were used as reference to create the Python rules dictionary within the main (Game) Module.

Within the TAEeG Game Module, Rules are stored in a Python dictionary (**rules_dict**), which maps each Rule identifier (e.g., "**Rule_006**") to a corresponding function, or method, within the **Game** class. This allows the addition or modification of rules without disrupting the overall architecture. For example, if a new rule is introduced requiring a specific action based on a card, it can be added to the dictionary and integrated with the existing game logic.

Figure 8. TAEeG Game Module (Source Data: Cards).
Semi-structured list of Karma and Bodhisattva cards and card contents (Appendix A: Table 5) were imported to an SQLite database, for reference within the Game Module.

Note that the **Game Module** makes calls to the **Card (DB) Module**, when avatars land on spaces that are associated with Rule_005. See the following section (3.3.2) for details. This module also incorporates turn management (see Section 3.3.4).

Finally, the **Game Module** updates the **GUI** dynamically to reflect changes in avatar location. As avatars progress along the Wheel of Training, movement is cyclical: as long as the avatar remains within a given path (outer, intermediate, or inner track), they wrap around to the beginning when they reach the end of the path. However, some karmic narratives trigger movement off the Dharmachakra, to one of the six cosmic realms (Heaven, Hell, etc.). In this case, avatars must move through all six realms and then return to their previous position on the game board. The Game Module references path_connections (Appendix A: Table 4), which are

statically coded in TAEeG version 1 (Figure 9). It uses this look-up information to update the avatar position and updates the GUI accordingly (Figure 6).

Figure 9. TAEeG Game Module (Source Data: Path Connections)
A table of "path connections" (Appendix A: Table 1) was used to hard code "ladders" (forward movement between tracks) and "chutes" (backward movement between tracks).

3.2.2.4. Database (DB) Module

The **Database (DB) Module** imports semi-structured (***card.xlsx***) data from the original (source) data file and supports real-time data retrieval to the main (Game) module. When a game initializes, the Pandas **to_sql** method is used to extract data from the original (source) data file (***card.xls***) and to check that all required fields are correctly formatted. Card data include karmic narratives and the associated rules for each Bodhisattva and Karma card (cf. Section 3.1.2 and Appendix A: Table 5). The verified data are subsequently stored in an SQLite database (**tae_egame_data**).

The DB module supports real-time data retrieval to the main (Game) module. When an avatar lands on a space that asks to draw a card, the Game module queries the DB to select a random card from one of the decks. Card contents are displayed in the GUI as popup messages using **Tkinter.messagebox.** Rules are triggered, and the game state is updated accordingly.

3.2.2.5. Graphical User Interface (GUI) Module

The GUI provides an interactive display, including a graphic layout that mimics the original TAE game board.

Figure 10. TAEeG GUI Module (Main Menu and Game Interface)
End user (Player) interacts with GUI display using a desktop computer and monitor (contact authors for detailed instructions on how to download, install, and run the software).

The Tkinter **Canvas** widget is used to create the game board layout, including the three circular Paths of Training and the central Hub (Figure 11). The **launch_game** method in the

Game class is used to set up the game's visual elements, including the concentric circles (Paths), individual spaces along each track, and cosmic realms.

Game Board Layout (Wheel of Training). To create the board, the script utilizes the **draw_circle** function, which takes in parameters for the canvas, coordinates, radius, outline color, and optional fill color. This function is repeatedly called to generate a series of nested circles with each representing different paths of the game. The largest circle forms the outer boundary, while progressively smaller circles define intermediary levels leading toward the center. The innermost circle (hub) is rendered in yellow.

Game Board Layout (Spaces). Spaces on the original TAE game board are represented in the 'space' table within the source data (excel) file (Fig. 11). In TAEeG, each space is assigned a coordinate within the **space_coordinates** dictionary. Using mathematical calculations involving trigonometric functions such as **math.radians**, **math.cos**, and **math.sin**, code within the GUI module defines evenly spaced positions along the circumference of each circle. The **draw_line** function is then called to create visual connections between these points (i.e., boundary lines delimiting each space).

Figure 11. TAEeG GUI Module (Source Data: Spaces).
A table of "spaces" (Appendix A: Table 3) was used to create the game board layout. Geometric coordinates were assigned to each space within the space_coordinates dictionary.

Certain spaces on the game board marked as "Bodhisattva Spaces" or "Karma Spaces." The Bodhisattva Spaces are rendered with a blue label: if a player avatar lands on one of these spaces, this triggers random selection of a Bodhisattva card from the database. Karma Spaces are rendered with a red label and trigger the random selection of a Karma cards.

Game Board Layout (Six Cosmic Realms). The TAEeG game board GUI also represents the Six Cosmic Realms from the original TAE game board (cf. Fig. 4). Each circle is rendered with a different fill color and labeled using Tkinter's **create_text** function. The placement of these circles is calculated using predefined coordinates.

Figure 12. TAEeG GUI Module (End-User Display).
Game interface is used to create end-user GUI display. Pop-up is used to display main menu. User accesses Main menu by clicking on "button" (icon in upper left corner of display).

Avatar Marker Position. A small blue circle is used to indicate the position of the avatar marker on the TAEeG virtual game board. When players initiate a virtual die roll at the start of each turn, the **move_player** method updates the avatar position based on the die roll result (current location + die_roll: see Sample Run-Through in Section 4 for examples).

The **create_player_piece** method is used to display the marker at the correct (current) coordinates when rules trigger updates in avatar location. The method considers the board's structure, which handles movement between different layers (i.e., the three circular tracks, the six cosmic realms, and the central hub). The script also accounts for rules that alter movement, such as "ladders" (special spaces or card draws that trigger movement in to the next innermost circle).

Avatar Movement (Die Rolls). The GUI module translates random die rolls into logical moves, keeping the avatar within the bounds of the board and updating its position accordingly. To display avatar movement, the algorithm calculates the avatar's new position based on the die roll. The avatar's position on the board is incremented by the rolled number, which is generated using a standard nine-sided die that produces a random value between 1 and 9 through a random number generator function. This new position is then validated by the code, to check that it does not exceed the board's boundaries. If the movement lands the avatar outside the board, specific conditions handle this, either by stopping the avatar at the end or looping back to the start, depending on the game rules. Whenever the avatar moves, the **canvas.coords** function updates the position of the marker within the canvas.

Player Interactions with GUI (Event Handling). The GUI layout is designed to be intuitive, with labeled controls that allow for gameplay actions, such as "Play," "Roll Die," or "Pause (or Resume) Game." The GUI handles these actions by responding to specific player inputs and triggering the appropriate functions. In particular, a play button, represented by a triangular shape, is located in the upper left corner of the UI and allows the player to start or stop the game. The simulated die roll is triggered when players click on a virtual "Die Roll" button. Also, a pause menu is implemented, which can be accessed by clicking on the play button's location. This menu provides options to resume the game, view the rules, or exit the session.

The event-handling algorithm supports player interactions with the GUI by responding to button clicks and player actions. It listens for specific inputs, such as rolling the die or resetting the game, and executes the relevant game logic based on these actions. This algorithm links player input to game responses.

A game state management algorithm monitors key game variables, such as the avatar's position and any special rules tied to specific board position. For example, if a cell has a special property (like moving the avatar forward or backward), the algorithm adjusts the avatar's position based on these rules. It checks for boundary conditions, such as reaching the center cell on the board, and triggers the end of the game when these are met. This algorithm continuously monitors the game's progression, updating the state as needed after each turn, which makes sure that the game logic aligns with the actions displayed on the GUI.

In addition, there is a pathfinding algorithm that is indirectly applied through the avatar's movement to provide a smooth visual transition as the avatar moves across the board. Although this is a simplified version of pathfinding (moving in a straight line rather than finding complex paths), it calculates and animates the movement across individual positions and improves the player experience by making the avatar's movement jump from one space to another to show progression within the game.

3.2.3. Turn Management

Turn management is handled within the Game Module. It tracks the number of turns a player gains, loses, or skips. It accomplishes this by updating turn counters (number of turns that the player must lose/skip) at the beginning and end of each turn. This feature helps manage complex sequences of events — e.g., where players are required to miss several turns and then perform another action associated with one of the rules.

3.2.4. Error Handling

To maintain data integrity and prevent gameplay interruptions, multiple error-handling mechanisms have been implemented. For example, missing data are accounted for by

incorporating fallback mechanisms that ensure the game continues smoothly when a specific card cannot be retrieved. If the queried card is not found, the system selects a default action rather than breaking the game logic.

Also, error-handling measures prevent invalid rule references from causing crashes. If a retrieved card contains a rule that does not exist in the rule dictionary, the system logs the issue and skips applying the effect rather than allowing the game to fail. Proper database connection management also contributes to data integrity because every time a query is executed, the system verifies that the database connection is correctly opened and closed to prevent memory leaks or performance breaks with syntax errors that violate the language's grammatical rules.

Alongside implementing structured data validation and checking that retrieved records work to the expected format, the system prevents inconsistencies that could otherwise lead to unpredictable gameplay behavior. Through these error-handling mechanisms, the game remains stable, with clear recovery procedures in place for potential data retrieval issues.

3.3. Testing and Debugging

The testing phase involved a combination of automated unit testing (3.3.1), manual gameplay trials (3.3.2), and iterative debugging to catch and resolve errors at different levels (3.3.3). The primary focus was to validate the correctness of the game’s movement system, rule application logic, GUI responsiveness, and event handling. Each of these components was systematically evaluated to confirm that it aligned with the design specifications and provided a smooth player experience. Given the complexity of the game, with its multiple layers of circular movement, space-based effects, rule-based modifications, and card-drawing mechanics, we performed iterative testing and refinement throughout the code development process.

3.3.1. Automated Unit Testing

Automated Unit testing was a crucial step in verifying the correctness of individual functions and making sure that each component worked as expected without raising any errors at player-based conditions. The movement function, **move_player**, was rigorously tested under various scenarios, such as standard movement and wrapping logic when transitioning between board sections, and special conditions like landing on a Bodhisattva or Karma space. Testing involved manually setting avatar positions and simulating die rolls to confirm that the expected outputs aligned with the actual game behavior. For example, when moving from space 64 to 73, the avatar needed to transition smoothly while following the intended circular path. The early tests showed that the transition calculations did not correctly adjust the avatar's placement, which caused irregular jumps across the board and sometimes completely off the board. After reviewing the logic and the math, the formulas were corrected to guarantee that any movement crossing a boundary properly adjusted the new position while maintaining visual continuity.

The unit testing also included testing how rules interacted with other game mechanics. Since many rules triggered position modifications, additional checks were performed. One common issue during unit testing was detecting off-by-one errors in movement calculations, which could lead to incorrect avatar placement. To address this, print statements were inserted to verify calculations at each step. Figure 12 shows one example.

```
if position in self.space_coordinates:
    x, y = self.space_coordinates[self.player.position]
    # Check if piece is initialized, if not, create it
    if self.player.piece is None:
        self.player.piece = self.canvas.create_oval(x - self.player.radius, y - self.player.radius,
                                                    x + self.player.radius, y + self.player.radius, fill="blue")
    else:
        self.canvas.coords(self.player.piece, x - self.player.radius, y - self.player.radius,
                           x + self.player.radius, y + self.player.radius)
else:
    messagebox.showerror("Error", f"Invalid starting position: {position}")
```

Figure 13. Code block illustrating unit testing.

By iterating through multiple test cases and adjusting logic where necessary, the unit testing phase played a fundamental role in establishing a stable foundation for the game's core mechanics.

3.3.2. Manual Gameplay Trials

While unit testing validated the correctness of individual components, manual gameplay testing focused on how the game functioned in real-world play sessions. Multiple playthroughs were conducted to observe the game's behavior under a variety of conditions. There was extra attention given to UI responsiveness because it had to be made sure that every avatar's action was correctly reflected on the game board, which could only be done through manual trial and error as everything on the board was drawn through calculations that were done by hand. For example, when rolling the die, the game needed to properly update the avatar's position, display relevant messages, and apply special space effects when applicable. One problem encountered during testing was that the game occasionally failed to update the avatar's visual position on the canvas, even though the internal position variable had changed. To debug this, a test was conducted where the die was rolled repeatedly while checking if the avatar piece visually moved on the board. The issue was ultimately traced to a missing canvas update call, which was resolved by adding: **self.canvas.update_idletasks()**. This function forces Tkinter to process all pending GUI updates, which makes sure to deliver all the changes immediately rather than waiting for the main event loop to cycle. This guarantees that when an avatar moves, the movement is instantly reflected on the board, which makes the gameplay feel smooth and responsive.

Beyond functionality, manual testing played an essential role in evaluating the overall player experience, which made sure that information was displayed clearly and that gameplay

flowed smoothly. Any inconsistencies in movement updates, missing UI feedback, or incorrect rule applications were logged and addressed. A particularly tricky bug was encountered where rolling a specific number while on a Bodhisattva space did not trigger the intended rule. When there were a number of set breakpoint in the **select_bodhi_card** function and printing the retrieved card details, it was discovered that some rule IDs in the database were not being correctly recognized in the **rules_dict** dictionary. The issue was resolved by implementing a function that checks to make sure that all the rule IDs were stored in a consistent format, which prevents mismatches between database entries and in-game logic.

3.3.3. Iterative Debugging

The debugging process was structured to isolate potential failures at every stage of gameplay. Since the game relied on dynamic updates to avatar position, rule management, and event-driven mechanics, errors came from incorrect movement calculations, rule conflicts, and unexpected player interactions. A major challenge was guaranteeing that the avatar's movement wrapped correctly across different circular sections of the board while maintaining the game's logical flow. One significant issue encountered early on was that avatars would sometimes move outside the designated circular paths when transitioning from one path section to another. This required a complete revision of the movement formula, which made sure that the avatar's position always adhered to the predefined space coordinates. When integrating a structured debugging approach, such as logging outputs and stack trace analysis, issues could be identified and systematically resolved. Debugging was about fixing problems and refining the codebase for better readability, efficiency, and long-term maintainability.

The debugging process was highly iterative, involving a combination of logging, step-by-step execution tracing, and interactive debugging tools. The logging statements were

strategically placed throughout the code to track key variables such as avatar position, die roll results, and triggered rules. These logs were valuable for tracing execution flow and identifying potential logic errors. For example, unexpected avatar movement in specific sections of the board was often traced back to incorrect condition checks, which were identified by reviewing log outputs and adjusting the logic accordingly. One effective debugging strategy was inserting **print** statements before and after key calculations to confirm expected behavior:

```
if self.is_lost_turn == True:
    self.lost_turn_counter -= 1
    if self.lost_turn_counter <= 0:
        self.is_lost_turn = False
        messagebox.showinfo("Turn Restored!", f"You can roll again")
    else:
        messagebox.showinfo("Lost Turn", f"Lost turn\n You still have {self.lost_turn_counter} turns left!")
        return
```

Figure 14. Code block illustrating steps counter.

This method helped identify discrepancies between the intended and actual movement results, which led to adjustments that secured consistent avatar progression across all board sections. Alongside manually tracking issues using print statements, debugging tools such as Python's built-in debugger and PyCharm's interactive debugging environment were used to examine game execution in real time. The ability to pause execution and inspect variable states allowed for a deeper understanding of program flow, which makes it easier to locate and resolve bugs. An extremely useful debugging technique involved simulating extreme conditions, such as rapid die rolls or prolonged game sessions, to stress-test the game's stability. During one test, rolling the die 50 times in quick succession caused an unexpected crash. When analyzing the error logs, it was revealed that the game attempted to access an invalid list index, which was corrected by providing all list references to be bounded within their valid range. Besides methodically addressing errors as they appeared and refining problematic code sections, iterative debugging played an integral role in guaranteeing that the game remained functional.

3.4. Versioning and Licensing of Software

GitHub was used for shared access to the TAEeG project resources and to store and manage different versions of the code. The ***/code*** repository contains three subdirectories:

1. **/old**: This directory includes all versions of the code *except for the current working version* (which is undergoing changes and validation testing), including early versions that may or may not be fully functional. Version numbers are incremented with each new version. The code with the highest version number in this folder (v1.0 at the time of writing) is equivalent to the code that within the ***/release*** directory.
2. **/release**: The code within the ***/release*** directory is the latest code that is fully functional and is available for download. It is equivalent to the code within the /old directory that has the highest version number.
3. **/working**: The code that is currently undergoing modifications is located in the ***/working*** directory. At the time of writing, this version is labeled as v1.1.

TAEeG Source Code is available on request. It is licensed under the GNU General Public License (GPLv3). Please contact us by email (gfrishkoff@gmail.com) for further information. The TAEeG Source Data contain copyrighted material from the original Training & Enlightenment© board game; please contact the Order of Buddhist Contemplatives to request access to these data.

4. Results

Contact authors for step-by-step instructions to install TAEeG (version 1) software, to setup and configure supporting applications, and to run the program.

In this section, we present UML diagrams to show the flow of procedures for TAEeG (v2). Figure 15 presents the first UML Sequence Diagram, showing interactions between the three code modules (GUI, Game, and DB) and the end user (Player).

Figure 15. UML Sequence Diagram #1. Illustrates basic interactions between the Player, GUI, Game (Avatar Movement & Rules), and Database (Cards) Modules.

Figure 16 presents a complementary diagram, showing interactions between the Player, GUI, Game, and Avatar modules. The list of time markers (t0 through t7) points to significant changes in the game state that result in changes to the GUI display. For example, at time t1, the player is prompted to enter a number between '1' and '8' to select a starting position (one of the 8 spokes of the wheel, representing 'Right Speech', 'Right Action', etc.). At time t2, the player is asked to press a key to initiate a simulated die roll. And so forth. In Section 4.2, we show a

specific sequence of game states (turns), representing a possible sequence of events for each of these time points, given the current (release) version of TAEeG.

Figure 16. UML Sequence Diagram #2. Shows sequence of actions within and between modules. Numbers on left side (t0–t8) mark key events resulting in changes to GUI display. Cf. following section for a sample "run through," with screenshots corresponding to each timepoint (Figs. 18–28).

The remainder of this section describes outcomes of this project, including fidelity of TAEeG (v1) to the design principles and software specifications outlined in Section 3, followed by a sample run through the program, illustrating a specific game play sequence, and accompanying notes.

4.1. TAEeG (v1): Sequential UML Diagrams

TAEeG (v1) meets the design specifications that were outlined in Section 3.1.4:

- Key contents of the analog (TAE) game, including the game board structure (locations), karmic narratives, and key variables that change over time (e.g., number of missed turns) were captured and stored in a digital (SQLite) database;
- The game interface (GUI) was implemented, tested, and validated using Tkinter, a python library for GUI development and execution; and
- The game mechanics were implemented using modular software design, to support iterative testing and refinement of the software, robust data management, and extensibility for future versions of the software.

TAEeG (v1) is fully functional and satisfies core design principles outlined in Section 3.1.5:

- The virtual game board faithfully recreates core elements of the original, analog board. It is designed as a circular, multi-layered structure with distinct paths and interactive spaces, which visually represent a journey through different realms. It consists of four concentric rings each with a different number of spaces. The outermost rings are outlined with clear boundary lines, which makes sure that movement is easily trackable. To maintain visual consistency, all spaces are equidistantly placed along the rings, which follow a precise angular calculation. This radial symmetry improves the board's structure and allows for smooth transitions between different layers of play. The outermost ring has 86 spaces, the second ring has 63 spaces, the third ring has 47 spaces, and the center of the board represents the state of enlightenment, which players can reach through specific game conditions. The spaces are arranged in a radial pattern, which provides smooth transitions between rings as players progress.

- The Game Board GUI (Fig. 17) is interactive and intuitive. In addition to the virtual game board, it includes a virtual die-roll button, which allows players to initiate die rolls, and a pause menu, which allows to resume or exit the game. It also includes pop-up notifications that inform the players of their progress, the effects of landing on specific spaces, and the consequences of drawing a card.

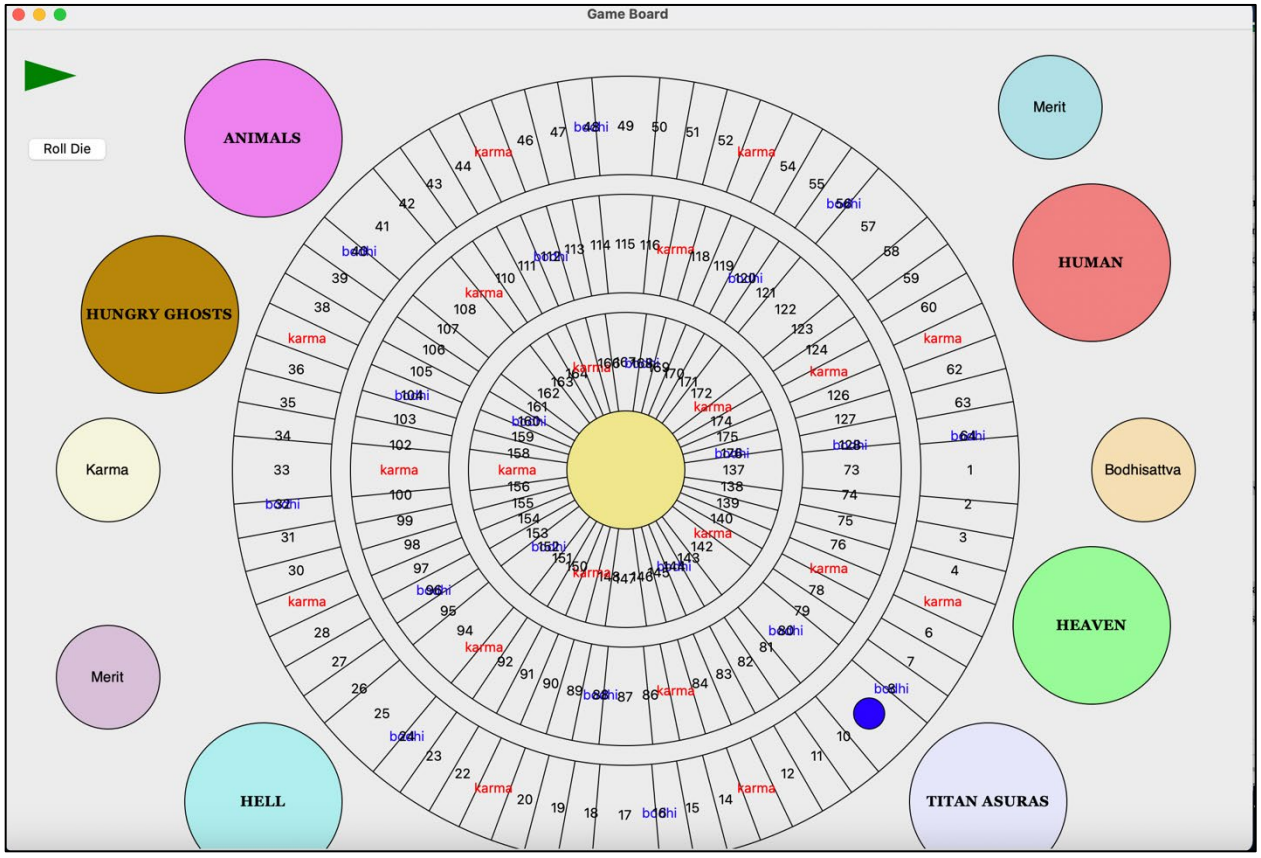

Figure 17. TAEeG (v1) Game Board GUI.
Blue dot (lower right), Avatar marker.
Green arrow (upper left), Main Menu (start, stop, pause game).
Roll Die button, initiates each Turn.

4.2. TAEeG (v1): Sample Run

This section illustrates how the TAEeG (v1) GUI appears when there are changes in the game state that result in updates to the interface. Each subsection includes a sequence of game states that illustrates one or more Rules (game mechanics).

Starting the Game (t0-t1). When the game launches, the Game Menu appears as a pop-up. The user clicks "Play Game" to start (Fig. 18; cf. **t0** in Fig. 16).

Figure 18. Sample Run (**t0**): TAEeG Game Menu (Pop-Up).
Cf. timepoint **t0** in Figure 16.

To enter the Wheel of Training, the user selects one of 8 spokes (branches of the Eightfold Path. For example, Space 8 (Fig. 19) corresponds to Right Action (cf. Figs. 1 & 53).

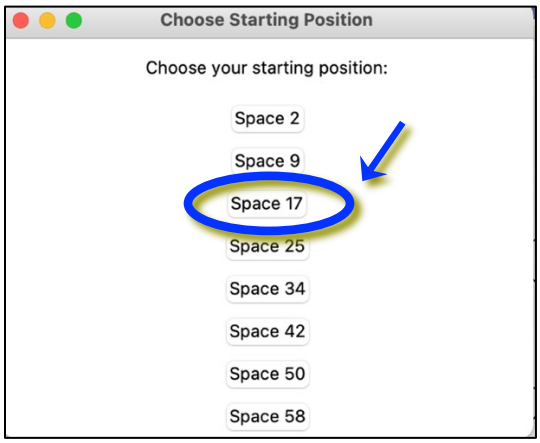

Figure 19. Sample Run (**t1**): TAEeG Path Branch Menu (Pop-Up).
User clicks "Space 17" (Path Branch = '**Right Effort**'). Cf. timepoint **t1** in Figure 16.

Movement along the Path (t2-t3). At the beginning of each turn, the User clicks "Roll Die" (Fig. 20; cf. **t2** in Fig. 16). This triggers a virtual die roll (pseudorandom selection of number between 0 and 8).

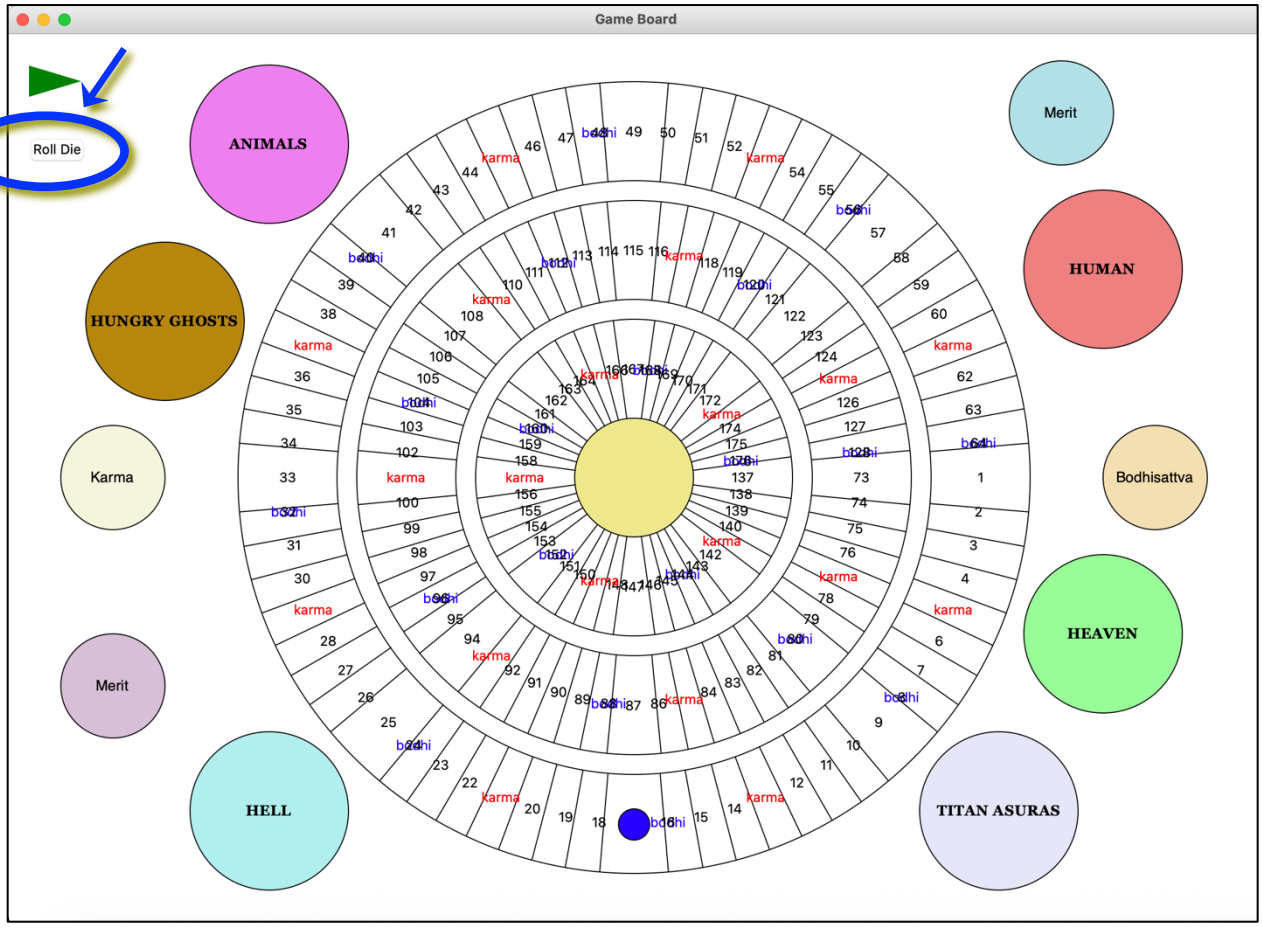

Figure 20. Sample Run (**t2**): Default GUI Game Display (start of each Turn).
Cf. timepoint **t2** in Figure 16.

After the die roll is initiated, a pop-up message appears, indicating the new target location. For example, in Figure 21 (after the die roll), the user is informed that the new target

location is space #22 (Fig 21: *left*)[7]. After the user clicks the button to initiate movement, their avatar marker is moved to space #22 and the die roll result is displayed (Fig 21: *right*).

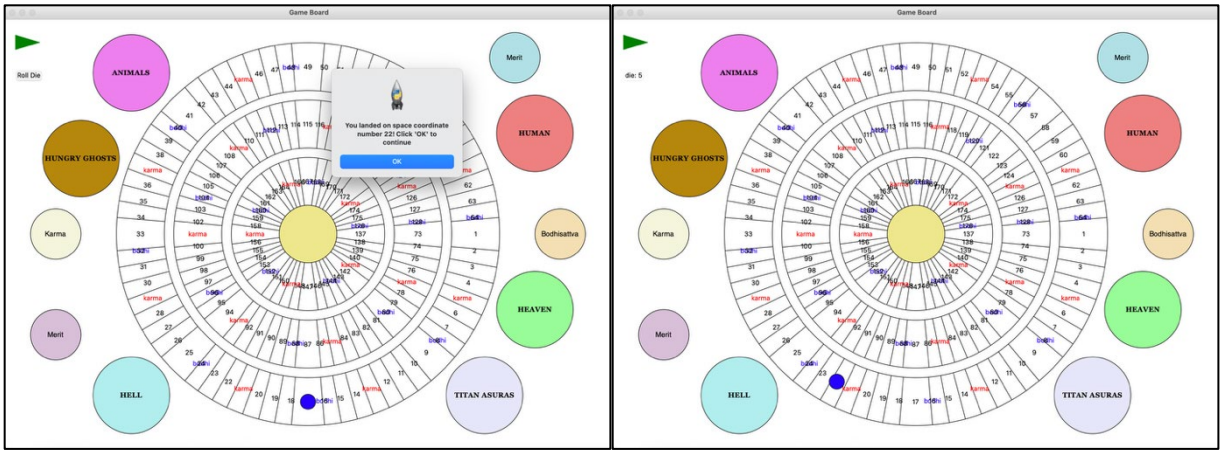

Figure 21. Sample Run (**t3**): Die roll results (new avatar location).
Left, pop-up message prompting user click to move avatar to new location.
Right, result of clicking 'ok' (update in avatar position marker).
Cf. timepoint **t3** in Figure 16.

Bodhisattva Card Space (t4-t6). If a player lands on a Bodhisattva Card space, the narrative content associated with the card is displayed in a pop-up message (Figs. 22-23; cf. **t4** in Fig. 16).

[7] The die roll results should be displayed along with the pop-up message to the user, prompting them to click "OK" to move to the new location. This feature will be updated in future versions.

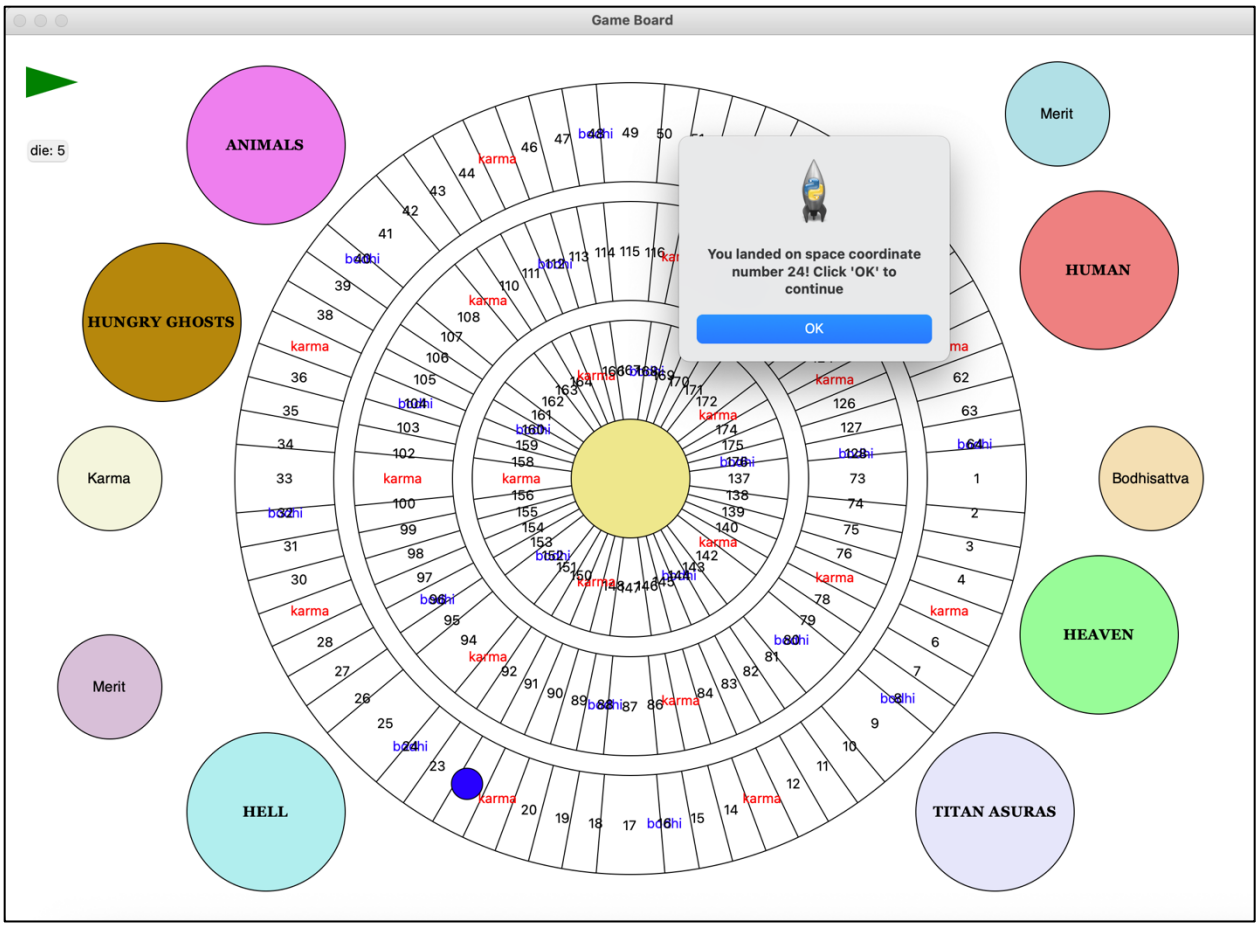

Figure 22. Sample Run (**t4**): Bodhisattva Card Draw.
Avatar lands on space #25, which triggers as Bodhisattva Card Draw.
Cf. timepoint **t4** in Figure 16.

In the following example (Fig. 23), the karmic narrative has the following structure:

- Karmic Action (*karma*): "Say that which is untrue."
- Karmic Consequence (*vipaka*): "Lose three turns."

The karmic action (Skt. *karma*) illustrates a "negative" (***unskillful)*** action of ***Speech*** (cf. Table 1). The karmic consequence (Skt. *vipaka*) is the prompt to sit still (i.e., ***meditate***) for three turns, before continuing game play.

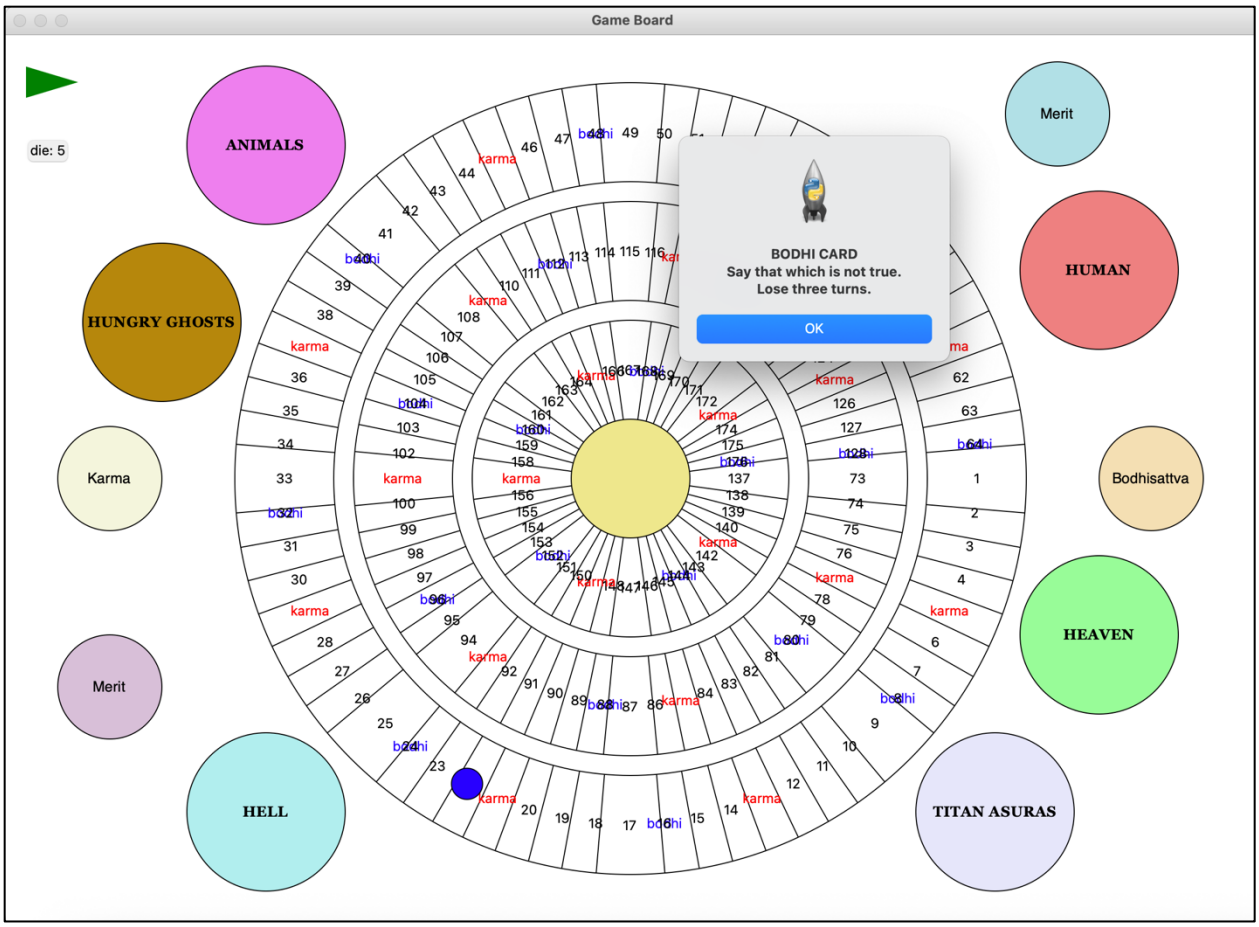

Figure 23. Sample Run (**t5**): Bodhisattva Karmic Narrative.
GUI displays karmic narrative and prompts user to click "OK" to trigger karmic consequences (Lost turns). Cf. timepoint **t5** in Figure 16.

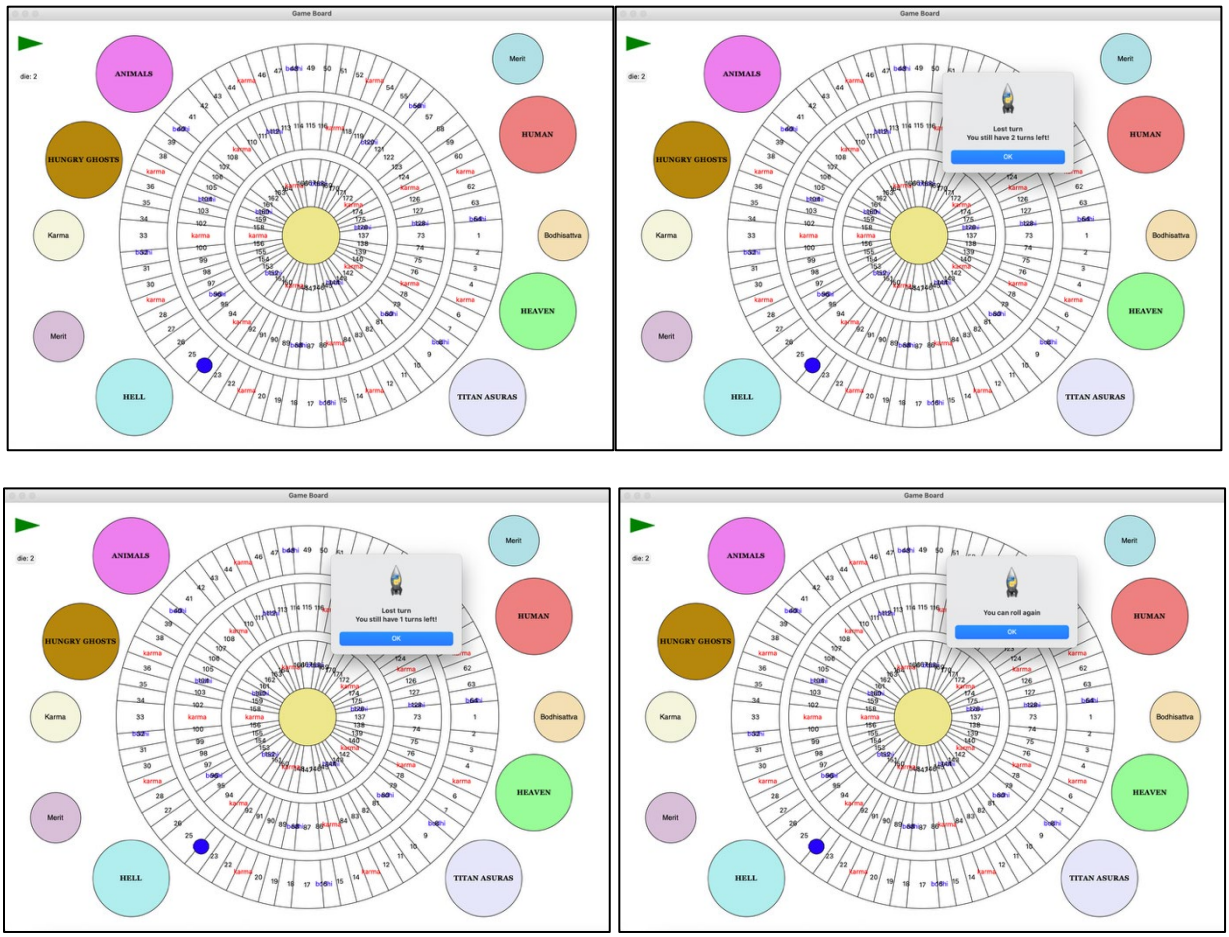

Figure 24. Sample Run (**t6**): Lost turns (karmic consequence of unskillful karma).
User is prompted to "sit still" (meditate) for next 3 turns.
Cf. timepoint **t6** in Figure 16.

This sample game play sequence illustrates Rule 008 ("lose XX turns"), which is associated with the ***Meditation*** pillar of practice (cf. Section 3.2.2.3).

Karma Card Space (t4-t6). If a player lands on a Karma Card space, the narrative content associated with the card is displayed in a pop-up. In the following example (Fig. 25), the narrative has the following structure (narrative not displayed in Figure):

- Karmic Action (*karma*): "Take a good look at yourself."
- Karmic Consequence (*vipaka*): "Go in to the next circle (Path)."

The karmic action (Skt. *karma*) illustrates a "positive" (***skillful)*** action of ***Mind*** (cf. Table 1): that is, the decision to consider one's attitude of mind and skillful and unskillful tendencies or habits. The karmic reward (i.e., ***merit***) is to move inwards to the next innermost Path (similar to accelerated movement up a ladder in Snakes & Ladders).

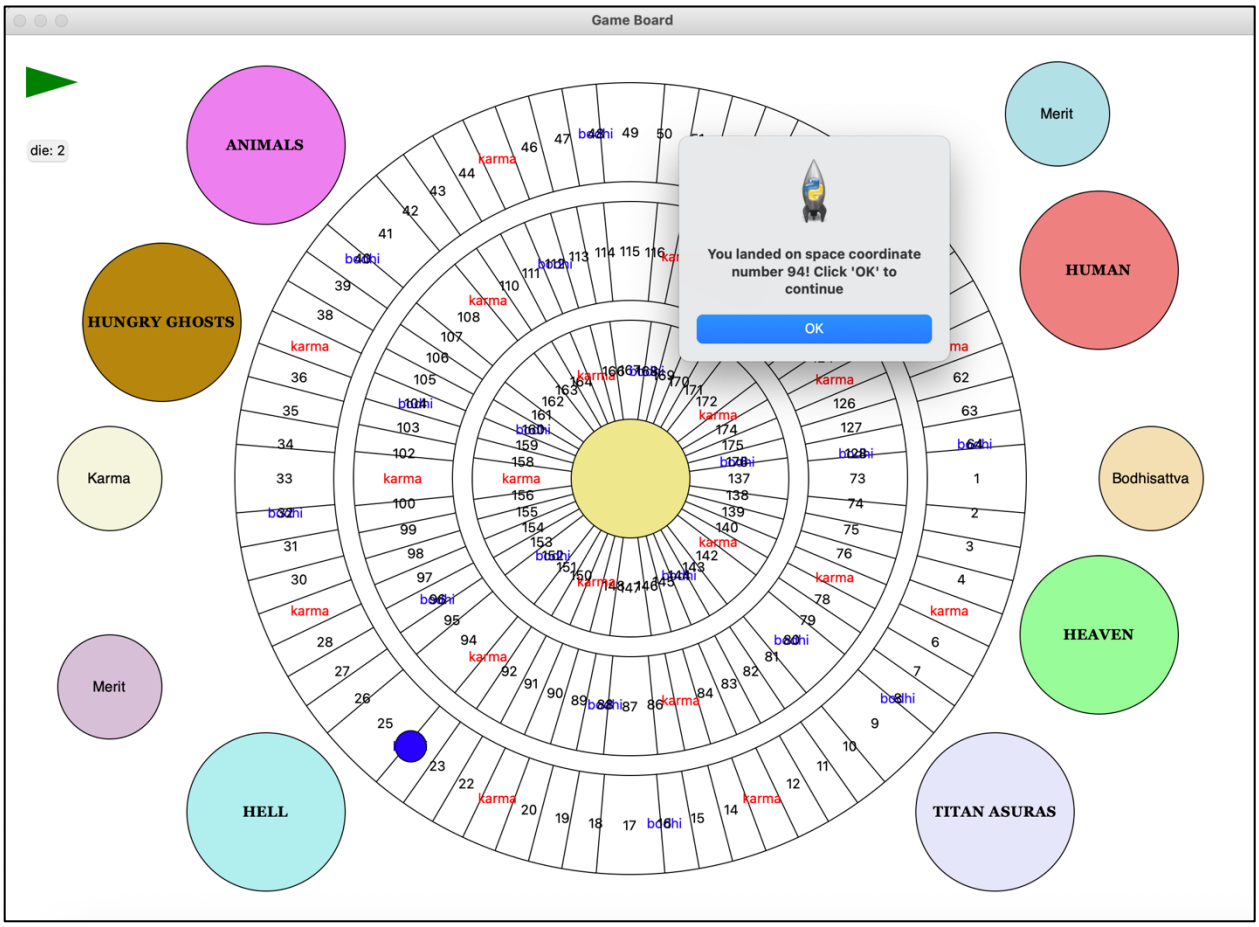

Figure 25. Sample Run (**t4**): Karma Card Draw.
Avatar lands on space #24, which triggers as Karma Card Draw.
Cf. timepoint **t4** in Figure 16.

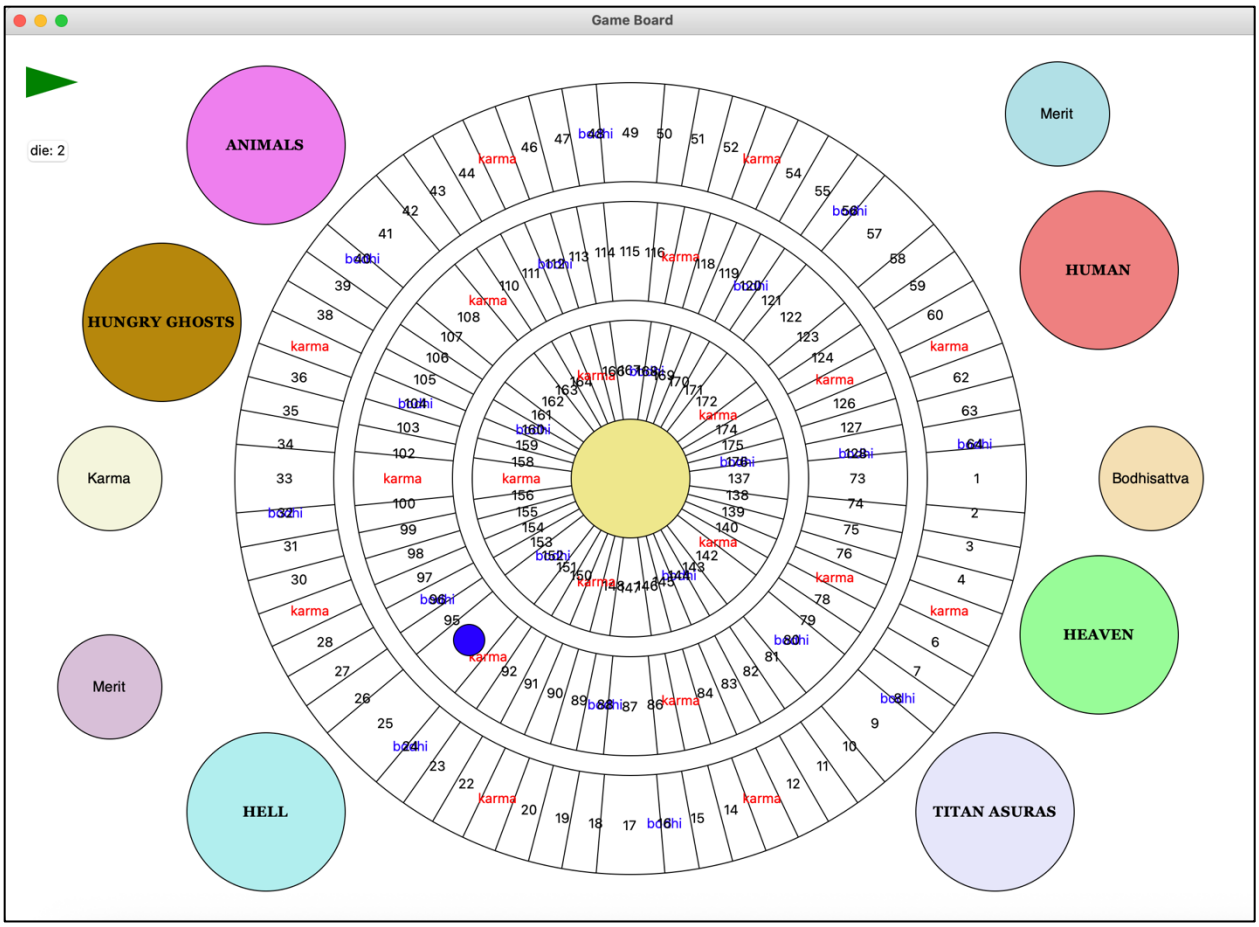

Figure 26. Sample Run (**t6**): Move inward to next Path (due to skillful karma).
Cf. timepoint **t6** in Figure 16.

This sample game play sequence illustrates Rule 005a (Karma Card draw) and Rule 006 ("go in one Path"), which is associated with the ***Ethical*** pillar of practice (cf. Section 3.2.2.3): skillful actions are rewarded by accelerated movement along the Path of Practice (cf. Ladders in "Snakes & Ladders" game; Section 2.2.1).

Spokes (Ladders)/Reaching the Center (t7-t8). If a player lands on a space that is connected to one of the eight spokes, they are automatically moved inward along the spoke (e.g., from the inner path to the center Hub, as illustrated Figs. 27-28). In this way, spokes function as ***Ladders***, which trigger accelerated movement along the Path of Practice.

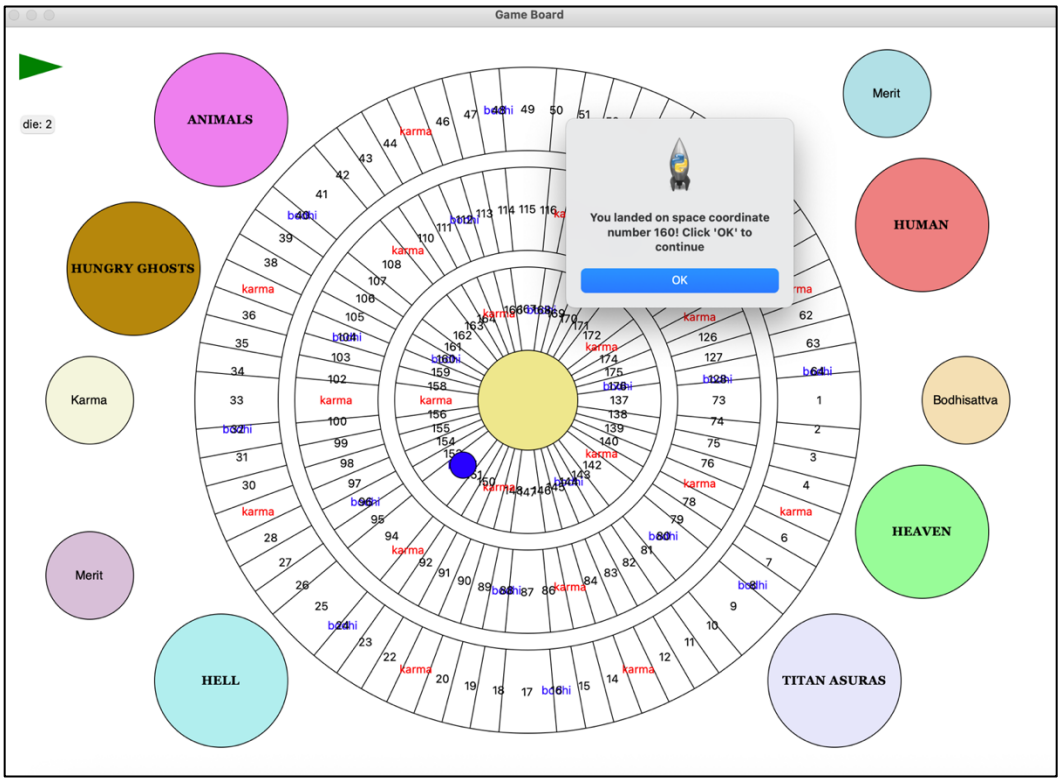

Figure 27. Sample Run (**t7**): Spoke (Ladder).
Avatar lands on one of the eight Spokes (Path Branches).
Cf. timepoint **t7** in Figure 16.

When the player reaches the center of board (yellow circle) (Fig. 28; cf. **t8** in Fig. 16), they have had a simulated *kensho* (Japanese for "glimpse of Enlightenment"; Jiyu-Kennett, 1989). In the original TAE game, the user is prompted to decide whether to remain in the center (exit the game) or to continue training (start over again at the beginning). In TAEeG v1, the game terminates when the player avatar reaches the center.

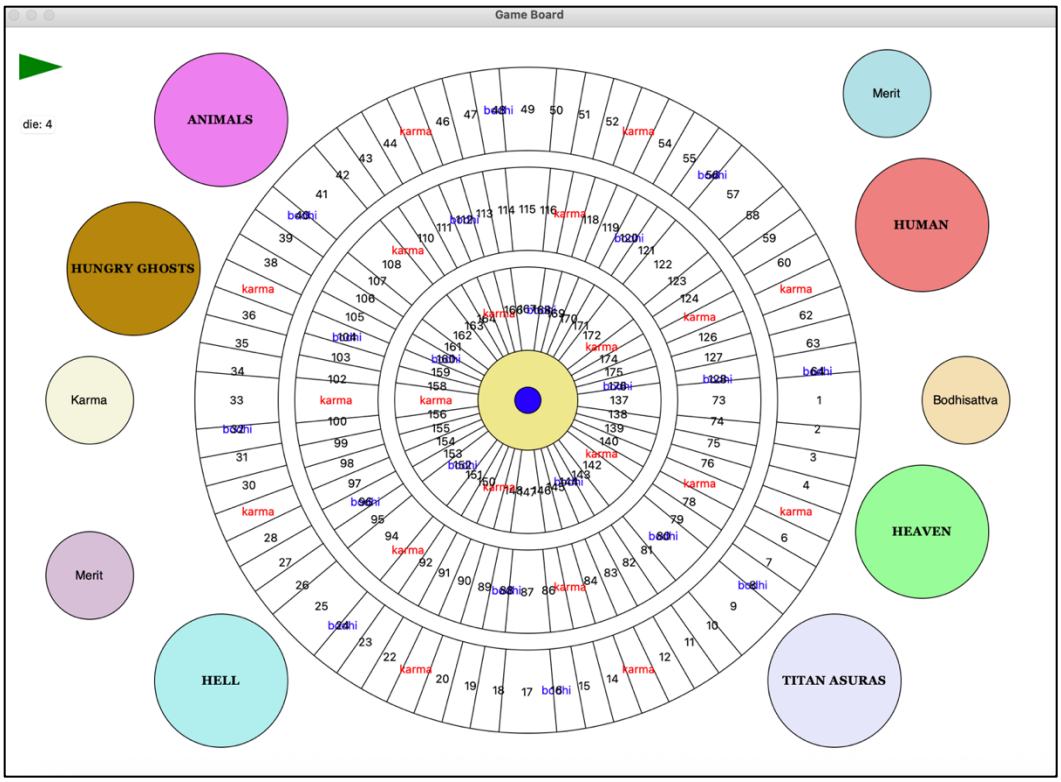

Figure 28. Sample Run (**t8**): Avatar reaches the Center (glimpse of ***Enlightenment***).
In TAEeG v1, the game terminates. In future versions, game play continues, and the player is prompted to decide whether to remain in the center (***nirvana***) or return to ***samsara*** (continue to train, to help support all beings to move towards enlightenment).
Cf. timepoint **t8** in Figure 16.

This sample game play sequence illustrates Rule 001 ("go in one Spoke"), which is associated with the ***Wisdom*** pillar of practice (cf. Section 3.2.2.3): movement inwards towards the center represents a deepening of embodied (procedural), as well as conceptual (declarative), understanding of how and why suffering arises, and how to develop skillful habits of Body, Speech, and Mind. The way of the Bodhisattva (according to Zen Buddhism) is continuous, selfless training: thus, the player who chooses to travel the Bodhisattva Path would choose to continue playing (see following section: "Future Directions").

5. Limitations and Future Directions

By design, the current (release) version of TAEeG excludes key features of the original board game (cf. Section 3.1.2). Section 5.1 reviews these features and discusses limitations of the methods and tools that were used for this first phase of code development.

Sections 5.2 and 5.3 outline future changes in the software to address these limitations. Future work is divided into two major phases. The first phase focuses on completing a full-featured single-player experience, while the second expands the game into a multi-player environment with advanced functionality and learning-focused features.

Finally, Section 5.4 considers future experiments to test specific Gaming and Learning components of TAEeG, as well as usability and efficacy studies, to evaluate performance (speed, efficiency of code execution), and learning and instructional outcomes.

5.1. Limitations

Throughout the development process, there were several limitations that affected the current implementation and future scalability of the game. One of the major challenges was the integration of complex game mechanics with the graphical user interface (GUI). TAEeG was designed with modular architecture, which requires the game logic, rules, and player interactions to work together. In practice, coordinating updates between these components often led to synchronization issues. For example, guaranteeing that player movement reflected accurately in the GUI, especially when handling special spaces or transitions between realms, demanded extensive debugging and refinement. These challenges were compounded by the need to dynamically update the visual representation of the board while maintaining consistent game state logic in the background.

Another significant challenge was implementing and debugging the wrapping logic for player movement across different sections of the board. The game requires players to navigate circular paths and transition between realms based on specific conditions. The movement algorithm had to account for edge cases, such as crossing boundaries or entering the six realms, which required careful planning and testing. The errors in these transitions sometimes caused the player icon to end up in the wrong places, which made repeated testing necessary to fix them.

Another limitation of the current implementation is its lack of scalability for multi-player functionality. The current design supports a single-player experience; extending it to accommodate multiple players would require significant modifications, especially in how player states, turns, and interactions are managed. The absence of multi-threading or asynchronous updates in the current architecture complicates the integration of real-time multi-player interactions: the GUI is dependent on Tkinter's main event loop, which does not support concurrent user inputs seamlessly.

Another limitation is the dependence on static data sources for cards and rules. The current version uses an SQLite database to store and retrieve card data. However, it does not allow for seamless updates to the source data. For example, adding new rules or modifying existing ones would require changes to the database and code directly, which reduces the flexibility to adapt the game post-deployment.

Finally, the current version includes basic visual and interactive elements that Tkinter supports. However, richer visualization may benefit from the use of a different development platform, such as Unity Game-Development Software, which is based on C++ and C#, instead of Python, and supports rendering of complex 2D and 3D graphics.

5.2. TAEeG (v1.5): Full-Featured, Single-Player Version

Version 1.5 will include interaction with a fully relational (SQL) database (Section 5.1.1), enhancements of the GUI, including added instructions to guide user interactions and attentional cuing, to support focused learning and engagement (Section 5.1.2), and added features from the original TAE (Section 5.1.3), which were excluded from TAEeG (v1) to streamline initial code development.

5.2.1. Relational (SQL) Database

In TAEeG_v1, an SQLite database is used to store and retrieve source data (e.g., see Appendix A: Table 5 for sample 'card' data and Table 6 for current list of 'rules'). This design does not allow for dynamic updates to source data. For example, adding new rules or modifying existing ones would require changes to the database and code directly, which reduces flexibility to adapt the game post-deployment.

Future versions will include a fully relational (SQL) Database and read/write functions that dynamically import source data (Appendix A) and update key variables on each turn. The use of a fully relational (SQL) database will make future versions much more flexible. For example, different karmic narratives could be loaded for different games, depending on learner preferences or progress along the path. Similarly, information that is currently stored in the Avatar class could be linked to the SQL database, to support dynamic storage and management of multiple avatars (multiple players), changes in avatar type and rank, accumulated points (merit), and player interactions (i.e., transfer of merit).

5.2.2. Enchacement of GUI

While the visual and interactive elements of the game are functional, they are constrained by the capabilities of Tkinter. Figure 53 shows a mock-up of an alternative, proposed GUI, with the following features:

- Simplified rendering of Path. Individual spaces are not labeled (or shown). Rather, information associated with the avatar's current position are displayed in labeled text boxes, which have fixed location and are updated with each change in the game state.
- Fixed layout with **labeled text boxes** to Instructions (*top*) and Narrative Content (*center left*: Karmic Context; *center right*: Karmic Consequence). These boxes can be dynamically populated from the SQL database (see Appendix A: Tables 1, 3 6, 7).
- **Attentional cuing** (highlighting individual elements with yellow boundary) to direct user attention to components, such as Die, Instructions, Karmic Context, etc.
- **Turn counters** (*left side*) that increment key metrics, such as avatar rank, current position, #metric points available and #turns to be skipped.
- **Legends** (*right side*) to remind players of key concepts, such as the Six Realms (top right) and the Noble Eightfold Path (bottom right).

Information from the turn counters can also be stored and tracked over time and displayed to users in a User Dashboard (Figure 54). These data can be used by learners and instructors to prompt deeper reflection on various aspects of game play and to inspire thoughts on how game play relates to their daily (zazen and mindfulness meditation) practice.

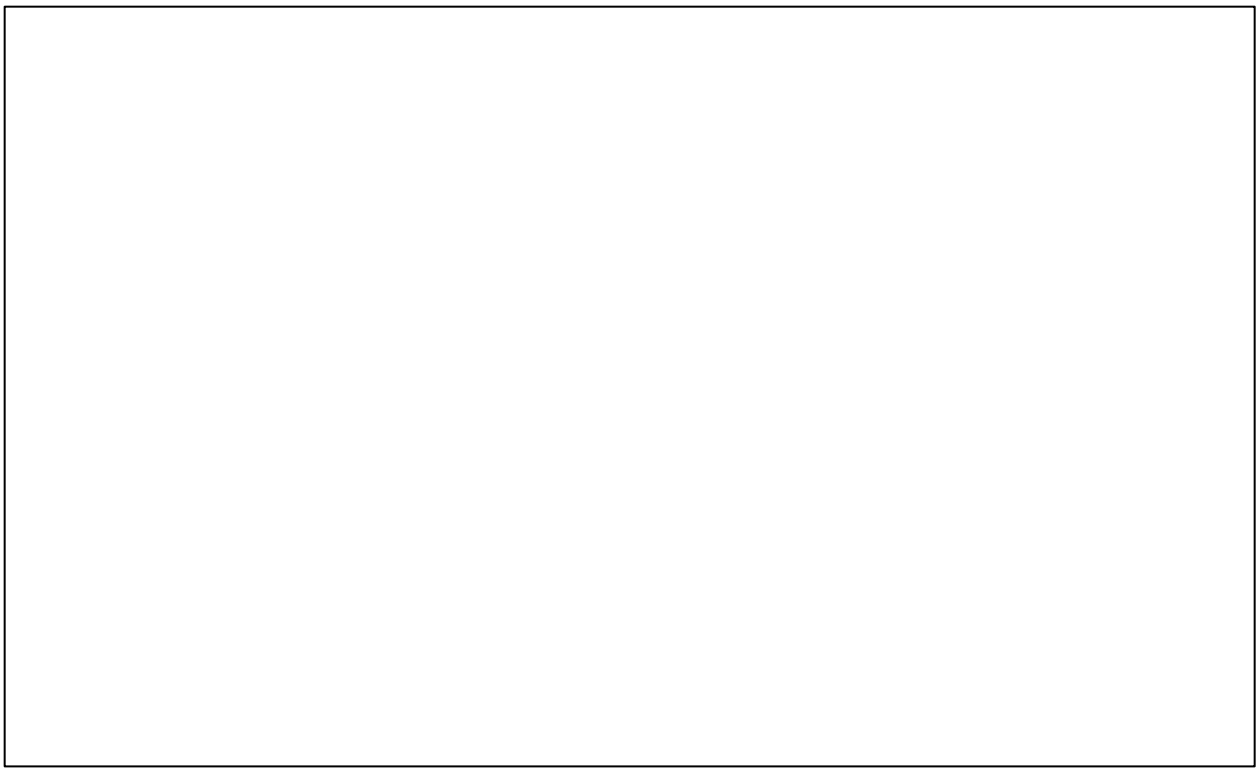

Figure 29: Mock-up of GUI for future versions of TAEeG.

A Game Play History, aggregating Player metrics overturns within each game session, could be displayed in a separate Dashboard (Fig 54: lower right). This information could be particularly helpful for Buddhist teachers and monastic guides, in providing feedback to students and suggestions on how to use insights from TAE to make adjustments in their daily practice.

Figure 30. Mock-up of Dashboard Design for future versions of TAEeG.
Trainee Dashboard (top right), showing Learning and Gaming metrics, aggregated overturns. This dashboard could be accessed at any point during game play.
Game Play History (bottom right*)*, showing summary metrics for each game session.

5.2.3. Added Features from Original TAE

In Version 1.5, the game will feature a complete set of rules to create a richer single-player experience. Avatar ranks will be introduced, which will provide a structured system for tracking player progression, and unlock additional gameplay mechanics (e.g., use of merit points). In addition, *merit points* (positive karma) will be added, to reproduce a key feature of the original board game. When a player reaches an avatar rank greater than 2, they will have the option to "cash in" their merit, either to help themselves or to help others. Transfer of Merit will

add a layer of complexity to the program; however, this feature is essential to represent the cooperative nature of Buddhist training and practice.

The movement system will be expanded to include rules for navigating the six cosmic realms, so that players will have structured paths to move between realms and return to their place on the game board after making a complete circuit.

Finally, when the players reach the center of the board, they will encounter a key feature of the original board game: they will be asked to either remain in the center (representing the *arahant path*, which effectively terminates game play) or continue playing (thereby choosing to follow the *bodhisattva path*) (Jiyu-Kennett, 1989). The decision to continue playing (i.e., to choose the Mahayana path of practice) will be rewarded by an increase in avatar rank and additional merit points, to replicate a key feature of the original board game.

5.3. TAEeG (v2.0): Multi-Player Version with Advanced Functionality

Version 2.0 will support multiple players, player interactions, and more advanced features, including personalization of karmic narratives and adaptive algorithms, which could adjust feedback and other instructional variables as learners progress through the game.

5.3.1. Multiple Players and Player Interactions

One limitation of the current implementation is its lack of scalability for multi-player functionality. The current game logic supports a single-player experience; extending it to accommodate multiple players would require significant modifications, especially in how player states, turns, and interactions are managed. The absence of multi-threading or asynchronous updates in the current architecture will present a challenge for implementation of real-time multi-

player interactions. An important factor is that the GUI is dependent on Tkinter's main event loop, which does not seamlessly support concurrent user inputs.

Version 2 and subsequent iterations will also introduce a fully relational (SQL) database. Python code will track updates in key variables (e.g., avatar rank, number of merit points, number of "lost" turns) on each turn, and additional code will save changes to the database.

Detailed tracking and saving of data will allow for several new features. First, the main graphical user interface (GUI) could display running totals for key turn variables (cf. Fig. 53). Second, it would support the creation of player dashboards (Figure 54): a User History Dashboard could display longitudinal performance metrics for individual players, and a Players Dashboard could summarize data across all users at specific moments in the game. This feature would allow players to track their progress during gameplay and review their history offline. The stored data could also be used to generate annotated "playback" videos, using content from the instruction tab in the source data file. These videos could serve as tutorials or instructional guides, which would provide the players with step-by-step explanations of game mechanics and strategies, either before or after gameplay.

A fully relational database, together with multi-player functionality, will support additional features. First, it will allow rotation among multiple users in a shared game session. Each player's data will be tracked individually, and the main GUI will display summary statistics for the active player at any given time. Also, rules will be introduced to allow the transfer of merit points (positive karma) between players. For example, when an active player faces negative karma, non-active players with a rank greater than 2 will be able to raise their hands and offer merit points to offset the penalty (cf. Box 1 in Section 2). This mechanism will encourage

player interaction and collaboration, which will be an enriching and fun aspect of the multiplayer experience.

Finally, future versions will incorporate features aimed at improving learning and accessibility. Instructional elements, such as attentional cues, clickable pop-up definitions, and help tips drawn from the instruction tab, will be added to clarify complex rules and increase the player understanding and engagement.

5.3.2. Personalization Options

Personalization is an advanced feature that would allow players to customize the game to align with their backgrounds and interests. One option is to create new karmic narratives that are appropriate for different demographics: e.g., ages (e.g., kids vs. adult), skill levels (e.g., learners who are new to Buddhism vs. advanced Buddhist practitioners), or domain of interests (e.g., healthcare, sports, family dynamics). Once these Karmic event sequences have been created, they could be labeled and associated with additional metadata is relevant to progress along the Buddhist path:

- **Path Branch** (e.g., narratives that are particularly relevant for practicing Right Speech, vs. those that are relevant for practicing Right Meditation)
- **Avatar type** (e.g., narratives that represent the arising of Greed vs. those that represent the arising of Anger or Ignorance)
- **Cosmic realm** (e.g., narratives that point to Hungry Ghost karma — such as addictive behaviors — vs. those that point to the Heavenly real — such as abundant leisure or wealth).

By labeling karmic narratives with these features, it will be possible to track player metrics, such as amount of practice with Right Speech or Right Meditation, and associate these metrics with learning outcomes, including *declarative knowledge* (e.g., ability to describe the Noble Eightfold Path) and *procedural skills and habits* (e.g., the amount of time spent "sitting still" during missed turns).

5.3.3. Increased Interactivity and Agency

TAEeG version 1 supports limited player interacts with the GUI. An updated GUI (e.g., Figure 53) could support more opportunities for players to reflect on narrative content, make decisions that reflect game play, and request additional information or help tips. This increased player agency, and interactivity should improve the user experience and result in more learning.

In TAEeG v1, each karmic narrative is presented all at once, as a pop-up message. In the revised GUI, narratives will be presented in three steps (cf. Appendix A: Tables 1 and 3):

- Step 1: Present ***karmic question***
- Step 2: Prompt user to select one of several options (***karmic action***)
- Step 3: Present ***karmic consequence***

Table 2 gives two examples from the original set of cards that were not used in TAEeG v1 (because they were too complex to implement in this version of the software).

Table 2. Complex Karmic Narratives (Jiyu-Kennett, 1989).

Karmic Question	**Karmic Options (*Karma*)**	**Karmic Outcomes (*Vipaka*)**
Are you impatient to get to the center?	Yes.	Draw a Karma Card.
	No.	Increase Avatar Rank.
	Can't decide.	Go to Hell.
Did you give away your merit points with the hope of a reward?	Yes.	Lose the rest of your Merit Points.
	No.	Receive another 3 Merit Points.
	Can't decide.	Lose 3 turns (opportunity to "sit still").

Note that the goal in presenting multiple response options is to increase user engagement and reflection on the nature of karma (ethical action); although there is a "correct" (or skillful) response in each case, it is not hard to guess which option is the more socially acceptable one. Thus, "accuracy" is not the goal. The main goal is to engage with the content.

Optional "Help Tips" can be displayed in one of several ways. One way is to allow users to click on different parts of the GUI layout, and to display information that is relevant at each point, referencing Look-up tables within the database, such as the Precepts table (Table 8) or the Instructions table (Table 9). An even more advanced option could make use of web scaping tools to retrieve information from a dedicated online resource, such as Sutta Central (https://suttacentral.net) or the Encyclopedia of Buddhism (https://encyclopediaofbuddhism.org).

With the introduction of a more complex GUI display and additional response options, it will be important to help the user navigate the display in a way that optimizes attention to, and engagement with, the relevant task components at each step. To this end, we will use attentional cuing: for example, when the GUI module is waiting for the user to read instructions and press "O.K.", the Instructions text box will be highlighted. Each highlighted component will be presented for a least 5 seconds: this requirement helps prevent users from mindlessly clicking through the game (aka "gaming the system").

5.3.4. Adaptive Features

Intelligent Tutoring Systems (ITS), such as Hodhod's AINS system (Hodhod, 2010) incorporate more advanced computational features that adapt scheduling and presentation of material based on user performance (cf. Section 2.1). Adaptive features include real-time feedback based on user responses (Frishkoff, et al, 2016) and scaffolding of easier and more

challenging tasks (e.g., presenting easier examples early in practice and more challenging material as the learner becomes faster and more skilled (Shute & Zapata-Rivera, 2012).

In TAEeG, narratives could present more or less challenging ethical dilemmas, which would provide for adaptive changes in selection of subsequent narratives, or changes in feedback, depending on player characteristics (e.g., Avatar Rank, time on task, proximity to center of wheel). Adaptive components increase the likelihood that players stay engaged over time, and (when properly implemented) they stimulate skill development and deeper learning. It is also worth noting that these more advanced features highlight a unique benefit of digitizing traditional (analog) games: scientific principles that have been discovered over the past two decades can be used to boost learning outcomes within the gaming environment.

5.4. Evaluation Studies

Future work should include studies that evaluate system performance (5.3.1), usability (5.3.2), and the impact of specific TAEeG Learning and Gaming Components on student outcomes (5.3.3).

5.4.1. Performance Evaluation

Performance analysis would provide quantitative and qualitative insights into the game's responsiveness, resource consumption, and scalability under varied operating conditions.

One important area of evaluation is the **graphical performance** of the game's interface. Since the game board is made using Tkinter's **Canvas** widget with mathematically calculated paths and dynamic updates, there is an important in assessing the rendering speed and frame rate under different system configurations is important because performance bottlenecks may appear as the game is scaled with additional graphical elements, animations, or multiplayer

functionality. Another factor is the **efficiency of data retrieval and database operations**. As the game currently uses SQLite for storing and accessing rule sets and card data, the future evaluation should measure the query execution times and database access latency, especially when the dataset grows with additional cards and rule entries. Although SQLite is efficient for local, single-user applications, benchmarking its performance under high-frequency access scenarios will help determine whether a transition to a more robust database system may be necessary in future versions. Also, **event-handling latency and GUI responsiveness** require detailed analysis. Since the gameplay is heavily driven by user interaction, such as die rolls, avatar movements, and pop-up notifications, evaluating how promptly the system processes these events under stress (e.g., rapid consecutive actions) will be important for a smooth user experience.

Moreover, **scalability testing** is something that should be taken into account, especially as the game expands to support multiplayer capabilities and networked gameplay in future updates. The current single-player model performs well in isolation, but a shift toward multi-avatar management will introduce new complexities such as concurrent actions, turn synchronization, and increased data loads. A key component of this next performance assessment would be simulating numerous players and monitoring how the system manages shared resources and simultaneous updates.

5.4.2. Usability Studies

Usability studies are conducted with real players and a variety of metrics, such as task completion time, player satisfaction ratings, and frequency of encountered bugs. These studies would help bridge the gap between technical performance and actual player experience.

Combining these realistic results with system-level performance metrics would offer a view of how well the game functions and where it needs improvement.

5.4.3. Controlled Experiments on Learning Outcomes

In future work, we hope to evaluate learning and practice outcomes when the game is played in its original (physical) form and in a digital (e-learning) format. We also intend to study the efficacy of TAEeG as a learning tool, by comparing outcomes for TAEeG vs. "business as usual" (Buddhist practice with and without TAEeG). Finally, with sufficient resources, we hope to conduct controlled experiments to test specific Learning and Gaming components (Carvahlo, et al., 2013), such as:

- Presence or absence of personalization
- Presence or absence of real-time feedback
- Presence or absence of help tips
- Scaffolding of easier vs. more challenging ethical scenarios

These studies would assess short- and long-term learning for each comparison, to determine the impact of specific gaming and instructional components. Study outcomes could inform the design and implementation of future systems.

6. Conclusion

This thesis presented the design and development of the *Training and Enlightenment eGame* (TAEeG), a digital implementation of the Zen Buddhist board game Training and Enlightenment™ (TAE), which was created by Reverend Master Jiyu-Kennett (1989) and manufactured and distributed by the Order of Buddhist Contemplatives (OBC).

TAE is a "serious," or educational, game: that is, in addition to being fun, it is designed to teach specific knowledge and skills that are helpful for Zen Buddhist learning and practice. The TAEeG project aimed to preserve key features of the analog (board) game, while exploring how this tool could be reimagined in a contemporary, digital context. The work combined technical software development with considerations of Zen Buddhist philosophy and training. The end product was a fully functional, single-player application that includes: a modular codebase, including an SQLite database to store karmic narratives, a set of algorithms to capture implicit rules (game mechanics), a movement module, and a detailed GUI layout, which updates dynamically in response to user input.

To our knowledge, TAEeG is the first educational Zen Buddhist e-game. It is designed to support procedural (skill-based), as well as declarative (conceptual), knowledge, which includes training and practice in Soto Zen meditation and Buddhist ethics. The current version provides a solid foundation for future work, which could include multiplayer functionality, changes in avatar rank, and more advanced features, such as personalization and adaptive instruction, which would build on recent developments in machine learning and other advanced computational methods for gamification and computer-based learning and instruction.

Appendix A: Source Data

TAEeG Source Data include copyrighted material from the original TAE© board game (Jiyu-Kennett, 1989). Appendix A provides snapshots of the semi-structured (excel) source data file that was used to capture the TAE game structure (including rules underlying game mechanics) and contents (e.g., structural and textual elements that express Soto Zen teachings). Please contact the Order of Buddhist Contemplatives (OBC) for information about the original board game and to request (restricted) access to the TAEeG Source Data.

TAEeG source data file (**TAE_Egame_data.xslx)** includes a 'space' tab (Table 3), with a list of spaces and associated rules (cf. Table 4: Rules). This information is hard coded within the Game Module, as described in Section 3. Other information associated with spaces in the original TAE game board — such as karmic narratives ("space context" and "space action" columns), and metadata (such as associated Path Branch) — are not implemented in TAEeG (v1).

Table 3. Spaces Data
(NOTE: p. 1 only)

Appendix A: Source Data (cont.)

The TAEeG source data file (**TAE_Egame_data.xslx)** also includes a 'path_connections' tab (Table 2). These data specify the paths that connect the spaces belong to outer, intermediate, and inner tracks, along with the Rules that trigger movement between tracks. For example, Rule 001 is triggered when players land on one of the spokes. This information is hard coded within the Game Module, as described in Section 3. Subsequent versions of TAEeG software will import this information from a relational (SQL) database, as described in Section 5.

Table 4. Path_Connections Data
(NOTE: p. 1 only)

Appendix A: Source Data (cont.)

The TAEeG source data file (**TAE_Egame_data.xslx)** includes a 'card' tab (Table 5), which includes a list of cards, card contents, and associated rules (cf. Table 6: Rules). These data are imported to an SQLite database and directly referenced in the Game Module, as described in Section 3.

Table 5. Card Data
(NOTE: p. 1 only)

Appendix A: Source Data (cont.)

The TAEeG source data file (**TAE_Egame_data.xslx)** also includes a ‘rule’ tab (Table 6), which lists pseudo-coded rules that are implicit in the original TAE game mechanics. The list of Rules is imported to, and stored in, the TAEeG SQLite database, along with Card data, from the Source Data File. See Section 3.2.2.3 (subsection titled “Rules”) for description of which rules where fully implemented in version 1.0.

Table 6. Rules Data

Appendix A: Source Data (cont.)

The TAEeG source data file (**TAE_Egame_data.xslx)** includes an 'Avatar' tab (Table 7). Avatar types and ranks are not referenced in TAEeG_v1, but will be implemented in subsequent versions. The image files associated with the 10 different avatar IDs are shown in Figure 55.

Table 7. Avatar Data
(NOTE: This Table Inactive in TAEeG_v1)

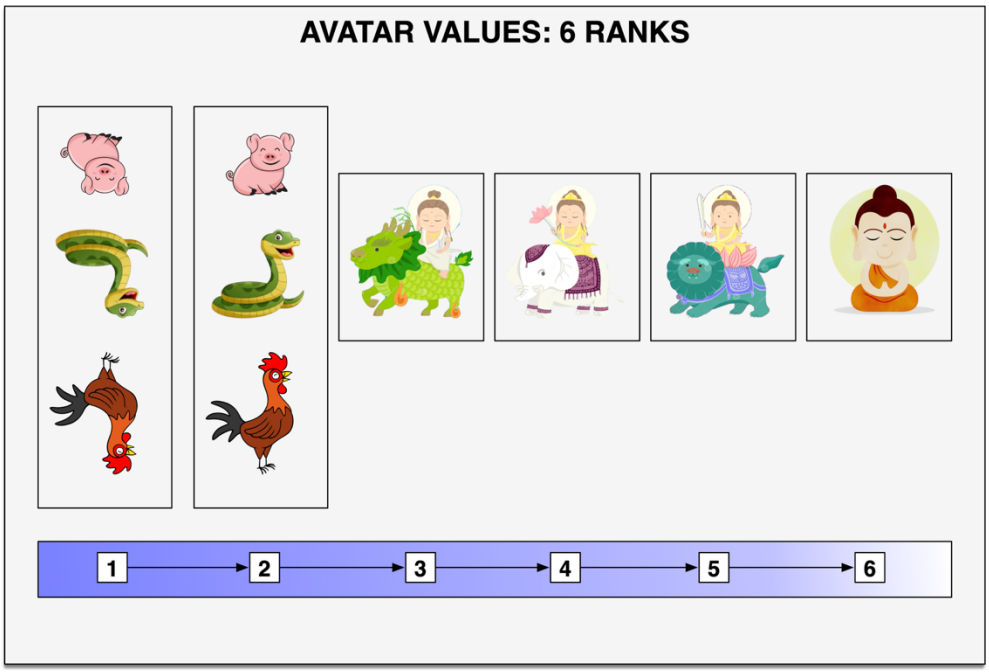

Figure 31: Cartoon depictions of Avatar Ranks in TAEeG (future versions). Based on the Avatar Ranks from the original TAE© board game (cf. Fig. 3).

Appendix A: Source Data (cont.)

Finally, the TAEeG source data file (**TAE_Egame_data.xslx),** includes a 'Precepts' tab (Table 8) and an 'Instruction' tab (Table 9). The precepts are based on (McPhillamy, 2016). Instructions are from the original TAE Game Instruction Manual or Quick Reference Guide (Jiyu-Kennett, 1989). Data from these two tables are not used in TAEeG_v1. In future versions, they will be displayed at the top of the GUI (see Fig 53), or as pop-ups (Definitions or Help Tips).

Table 8. Bodhisattva (Zen Buddhist) Precepts
(NOTE: This Table Inactive in TAEeG_v1)

Table 9. TAE Instructions and Help Tips
(NOTE: p. 1 only; This Table Inactive in TAEeG_v1)

Appendix B: Software Code

TAEeG began development in early 2024 as a digital adaptation of the original TAE board game. The goal was to translate the game's spiritual and karmic progression mechanics into an interactive single-player experience. Version 1.0 is the first prototype and represents a limited implementation of core features. TAEeG Source Code is licensed under the GNU General Public License (GPLv3). Please contact us by email (gfrishkoff@gmail.com) for access to the code and for additional information and support.

Creation Date: March 2024

Date Last Revised: May 27, 2025

Version Number: 1.0

```
####### taeeg_v1 #########

import tkinter as tk
import tkinter.messagebox as messagebox
import random
import math
from tokenize import Special
import bodhi_karma_card_grabs
from bodhi_karma_card_grabs import select_bodhi_card

import pandas as pd
import sqlite3
import tkinter.messagebox as messagebox

def draw_circle(canvas, x, y, radius, outline_color="black", fill_color=None):
    canvas.create_oval(x - radius, y - radius, x + radius, y + radius, outline=outline_color,
fill=fill_color)


def draw_line(canvas, x1, y1, x2, y2, color="black"):
    canvas.create_line(x1, y1, x2, y2, fill=color)


def draw_play_button(canvas, x, y, size, color="black"):
    half_size = size // 2
```

```
    triangle_height = int(size * 0.6)
    canvas.create_polygon(x - half_size, y - triangle_height // 2,
                          x - half_size, y + triangle_height // 2,
                          x + half_size, y,
                          fill=color)

class Player:
    def __init__(self, name, starting_position=2000, radius=15):
        self.name = name
        self.position = starting_position
        self.prev_position = None
        self.radius = radius
        self.piece = None

class Game:
    def __init__(self, root):
        self.root = root
        self.create_buttons()
        self.canvas = None
        self.space_coordinates = {}
        self.bodhi_spaces = []
        self.karma_spaces = []
        self.player = Player("Player1")  # Create a Player instance
        self.is_lost_turn = False
        self.lost_turn_counter = 0

        # t5: Apply specific rule effect (warp, loop, lost turn, etc.)
        self.rules_dict = {
            "Rule_006": self.rule_006, "Rule_016": self.rule_016, "Rule_004": self.rule_004,
"Rule_008a": self.rule_008,
            "Rule_008b": self.rule_008, "Rule_008c": self.rule_008,  "Rule_010a": self.rule_010,
"Rule_010b": self.rule_010,
             "Rule_010c": self.rule_010,  "Rule_010d": self.rule_010,  "Rule_010e": self.rule_010,
"Rule_010f": self.rule_010
        }

        self.card_action_nums_dict = {"one": 1, "two": 2, "three": 3, "four": 4, "five": 5,
                                      "six": 6, "seven": 7, "eight": 8}

        self.realms_dict = {"Hell": 2504, "Hell.": 2504, "Titan Asure": 2500, "Animals": 2502,
"Animal": 2502, "Hungry Ghost": 2503,
                            "Heaven": 2501, "Human": 2000}

    def create_pause_menu(self):
        pause_menu = tk.Toplevel()
        pause_menu.title("Pause Menu")
```

```
        pause_menu.geometry("400x300")

        resume_button = tk.Button(pause_menu, text="Resume", command=pause_menu.destroy)
        resume_button.pack(pady=10)

        rules_button = tk.Button(pause_menu, text="Rules")
        rules_button.pack(pady=10)

        main_menu_button = tk.Button(pause_menu, text="Exit Game", command=lambda:

self.exit_game(pause_menu))
        main_menu_button.pack(pady=10)

        pause_menu.mainloop()

    def exit_game(self, pause_menu):
        pause_menu.destroy()
        self.root.destroy()

    #the die roll
    def roll_die(self, die_button):
        if self.is_lost_turn == True:
            self.lost_turn_counter -= 1
            if self.lost_turn_counter <= 0:
                self.is_lost_turn = False
                messagebox.showinfo("Turn Restored!", f"You can roll again")
            else:
                messagebox.showinfo("Lost Turn", f"Lost turn\n You still have
{self.lost_turn_counter} turns left!")
                return

        die_number = random.randint(1,9)
        die_button.config(text=f'die: {die_number}')
        self.move_player(die_number)

    # Display the player's position
    def display_space_coordinate(self):
        if self.player.position in self.space_coordinates:
            space_coordinate = self.player.position  # Coordinate number of the space
            messagebox.showinfo("Landed on Space", f"You landed on space coordinate number
{space_coordinate}!")


    # t6: Check for enlightenment space and prompt player
    def reach_center(self):
        if self.player.position in [186]:
```

```
        # t7: At center — prompt player to 'Stay' or 'Continue'
        messagebox.showinfo("reached enlightenment", "reached enlightenment would you like to continue the game or stay here?")

    # t2: Die roll triggers random move and avatar position update
    #player movement with the rules
    def move_player(self, steps):

        self.prev_position = self.player.position
        print("at start of func", self.prev_position)

        print(f'Initial position: {self.player.position}, Steps: {steps}')

        if self.player.position <= 64 and self.player.position < 73:
            self.player.position = (self.player.position + steps) % 64 #% len(space_coordinates)
            print(f'After moving: {self.player.position}')
        if self.player.position > 73 and self.player.position < 128:
            self.player.position = (self.player.position + steps)
        if self.player.position > 128 and self.player.position < 137:
            print(((self.player.position + steps) - 128) + 73 )
            self.player.position = (((self.player.position + steps) - 128) + 73) - steps
        if self.player.position > 137:
            self.player.position = (self.player.position + steps)
        if self.player.position > 176 and self.player.position < 186:
            self.player.position = (((self.player.position + steps) - 176)+ 137) - steps
        if self.player.position >= 2500:
            self.choose_starting_position()

        else:
            print("issue with wrapping invalid space")

        print("after reposition start of func", self.prev_position)


        x, y = self.space_coordinates[self.player.position]
        self.canvas.coords(self.player.piece, x - self.player.radius, y - self.player.radius, x + self.player.radius, y + self.player.radius)


        print(f'After wrapping: {self.player.position}')

        special_spaces = [8,9,17,25,33,41,49,57,
                          73,80,87,94,101,108,115,122]

        just_move_in_circle = False
```

```
        # List of special spaces and their adjustments
        special_spaces = {
            8: 71, 16: 70, 25: 68, 33: 67,
            41: 66, 49: 65, 57: 64, 73: 57,
            80: 56, 87: 55, 94: 54, 101: 53,
            108: 52, 115: 51, 122: 50,

            166: 20, 173: 13, 138: 48, 141: 45, 148: 38, 153: 33, 158: 28, 163: 23
        }


        if self.player.position in special_spaces and self.prev_position < 2500:
            self.prev_position = self.player.position
            print(f"Hit special space at {self.player.position}")
            self.player.position += special_spaces[self.player.position]
            print(f"After special space adjustment: {self.player.position}")
            just_move_in_circle = True

        print("after special space prev space:", self.prev_position)
        print("current space after special:", self.player.position)

        x, y = self.space_coordinates[self.player.position]
        self.canvas.coords(self.player.piece, x - self.player.radius, y - self.player.radius, x +
self.player.radius, y + self.player.radius)

        # Call the function to display the popup message after the player moves
        self.display_space_coordinate()

        # t3: Card space → Show karmic context
        if self.player.position in self.bodhi_spaces:
            self.select_bodhi_card()
            print("bodhi")

        # t3: Card space → Show karmic context
        if self.player.position in self.karma_spaces:
            self.select_karma_card()
            print('karma')

        self.reach_center()


    #rules

    def rule_004(self):
        """
        move backwards
```

```
        """

        self.move_player(-10)

    def rule_006(self):
        """
        go in one cirle
        """

        print("made it to rule_006")
        self.move_player(15)

    def rule_008(self, turns_lost):
        """
        lost turn
        """

        self.is_lost_turn = True
        self.lost_turn_counter = turns_lost

    def rule_016(self):
        """
        go to center
        """

        self.player.position = 186
        x, y = self.space_coordinates[self.player.position]
        self.canvas.coords(self.player.piece, x - self.player.radius, y - self.player.radius, x +
self.player.radius, y + self.player.radius)

    def rule_010(self, realm):
        """
        go to hell
        """

        self.player.position = realm
        x, y = self.space_coordinates[self.player.position]
        self.canvas.coords(self.player.piece, x - self.player.radius, y - self.player.radius, x +
self.player.radius, y + self.player.radius)

    def select_bodhi_card(self):
        # Load the Excel file
        excel_file_path =
'/Users/nithi/Documents/GitHub/tae_egame/database/resources/data/Source Data
XLSX/cards.xlsx'
        df = pd.read_excel(excel_file_path)  # Load the sheet into a DataFrame
```

```
# Connect to SQLite database
conn = sqlite3.connect('TAE_Egame_data.db')

# Write the DataFrame to SQLite database
df.to_sql('tae_egame_data', conn, if_exists='replace', index=False)

# SQL query to select a random Bodhi card context
query = """
SELECT card_context, rule_ID, card_contents, card_action
FROM tae_egame_data
WHERE card_ID BETWEEN 'card_031' AND 'card_068'
ORDER BY RANDOM()
LIMIT 1;
"""

# Execute the query
cursor = conn.cursor()
cursor.execute(query)
result = cursor.fetchone()

# Close the connection
cursor.close()
conn.close()

# Print the card context if a card was found
if result:
    print("Rule ID:", result[1])

    # t4: Show karmic consequence and apply rule (movement, skip turn, realm warp, etc.)
    messagebox.showinfo("BODHI CARD", f"BODHI CARD\n {result[0]}\n {result[3]}")

    rule_ID = result[1]


    split_card_actions = result[3].split()
    card_action_flag = False
    card_action_realm_flag = False

    for i in split_card_actions:
        print(i)
        if i in self.card_action_nums_dict:
            card_action_num = self.card_action_nums_dict[i]
            card_action_flag = True
            print("made it to nums_dict")
```

```
            if i in self.realms_dict:
                realm = self.realms_dict[i]
                card_action_realm_flag = True
                print("made it here realms dict")

        if rule_ID in self.rules_dict:
            if card_action_flag == True:
                print("card_action_num condition")
                print(card_action_num)
                self.rules_dict[rule_ID](card_action_num)
            if card_action_realm_flag == True:
                print(realm)
                self.rules_dict[rule_ID](realm)
            else:
                print("before rule func call in bodie_selct")
                self.rules_dict[rule_ID]()
                print("im here G")

    else:
        print("No cards found in the specified range.")

def select_karma_card(self):
    # Load the Excel
    excel_file_path =
'/Users/nithi/Documents/GitHub/tae_egame/database/resources/data/Source Data
XLSX/cards.xlsx'
    df = pd.read_excel(excel_file_path)  # Load the sheet into a DataFrame

    # Connect to SQLite database
    conn = sqlite3.connect('TAE_Egame_data.db')

    # Write the DataFrame to SQLite database
    df.to_sql('tae_egame_data', conn, if_exists='replace', index=False)

    # SQL query to select a random Bodhi card context
    query = """
    SELECT card_context, rule_ID, card_contents, card_action
    FROM tae_egame_data
    WHERE card_ID BETWEEN 'card_001' AND 'card_030'
    ORDER BY RANDOM()
    LIMIT 1;
    """

    # Execute the query
    cursor = conn.cursor()
    cursor.execute(query)
```

```
        result = cursor.fetchone()

        # Close the connection
        cursor.close()
        conn.close()

        # Print the card context if a card was found
        if result:

            print("Random Karma Card Context:", result[0])
            print("Rule ID:", result[1])

            # t4: Show karmic consequence and apply rule (movement, skip turn, realm warp,
etc.)
            messagebox.showinfo("KARMA CARD", f"KARMA CARD\n {result[0]}\n
{result[3]}")

            rule_ID = result[1]

            split_card_actions = result[3].split()
            card_action_flag = False
            card_action_realm_flag = False

            for i in split_card_actions:
                print(i)
                if i in self.card_action_nums_dict:
                    card_action_num = self.card_action_nums_dict[i]
                    card_action_flag = True
                    print("made it to nums_dict")
                if i in self.realms_dict:
                    realm = self.realms_dict[i]
                    card_action_realm_flag = True
                    print("made it here realms dict")

            if rule_ID in self.rules_dict:
                if card_action_flag == True:
                    print("card_action_num condition")
                    print(card_action_num)
                    self.rules_dict[rule_ID](card_action_num)
                if card_action_realm_flag == True:
                    print(realm)
                    self.rules_dict[rule_ID](realm)
                else:
                    print("before rule func call in bodie_selct")
                    self.rules_dict[rule_ID]()
                    print("im here G")
```

```
    # t1: Prompt for and handle avatar starting location
    def choose_starting_position(self):
        start_menu = tk.Toplevel()
        start_menu.title("Choose Starting Position")
        start_menu.geometry("400x300")

        # Bring the start_menu window to the foreground
        start_menu.attributes('-topmost', True)
        start_menu.lift()

        tk.Label(start_menu, text="Choose your starting position:").pack(pady=10)

        starting_positions = [1, 8, 16, 24, 33, 41, 49, 57]

        for pos in starting_positions:
            button = tk.Button(start_menu, text=f"Space {pos}", command=lambda p=pos:
self.set_starting_position(p, start_menu))
            button.pack(pady=5)

        self.canvas.wait_window(start_menu)

    def set_starting_position(self, position, start_menu):
        self.player.position = position

        if position in self.space_coordinates:
            x, y = self.space_coordinates[self.player.position]
            # Check if piece is initialized, if not, create it
            if self.player.piece is None:
                self.player.piece = self.canvas.create_oval(x - self.player.radius, y - self.player.radius,
                                                            x + self.player.radius, y + self.player.radius, fill="blue")
            else:
                self.canvas.coords(self.player.piece, x - self.player.radius, y - self.player.radius,
                                   x + self.player.radius, y + self.player.radius)
        else:
            messagebox.showerror("Error", f"Invalid starting position: {position}")

        start_menu.destroy()

    # t0: Player clicks 'Play Game' → Launch game board and prompt for start position
    def launch_game(self):
        self.root.destroy()

        self.player.position = 2000
        self.player.radius = 15
```

```
        self.canvas = tk.Tk()
        self.canvas.title("Game Board")

        canvas_width = 1200
        canvas_height = 850
        self.canvas = tk.Canvas(self.canvas, width=canvas_width, height=canvas_height)
        self.canvas.pack()

        center_x = canvas_width // 2
        center_y = canvas_height // 2
        outer_circle_radius = 190 * 2

        inner_circle_1 = outer_circle_radius * 0.7
        inner_circle_2 = outer_circle_radius * 0.4
        inner_circle_3 = outer_circle_radius * 0.15

        buffer_circle_1 = outer_circle_radius * 0.75
        buffer_circle_2 = outer_circle_radius * 0.45

        world_circle = outer_circle_radius * 0.20

        draw_circle(self.canvas, center_x, center_y, outer_circle_radius)
        draw_circle(self.canvas, center_x, center_y, inner_circle_1)
        draw_circle(self.canvas, center_x, center_y, inner_circle_2)
        draw_circle(self.canvas, center_x, center_y, inner_circle_3, fill_color="khaki")

        draw_circle(self.canvas, center_x, center_y, buffer_circle_1)
        draw_circle(self.canvas, center_x, center_y, buffer_circle_2)

        circle_positions = [
        (center_x + 450, center_y - 200),
        (center_x + 350, center_y + 320),
        (center_x + 450, center_y + 150),
        (center_x - 350, center_y - 320),
        (center_x - 450, center_y - 150),
        (center_x - 350, center_y + 320)
        ]
        circle_colors = ["lightcoral", "lavender", "palegreen", "violet", "darkgoldenrod",
"paleturquoise"]
        circle_labels = ["HUMAN", "TITAN ASURAS", "HEAVEN", "ANIMALS", "HUNGRY
GHOSTS", "HELL"]

        for (x, y), color, label in zip(circle_positions, circle_colors, circle_labels):
            draw_circle(self.canvas, x, y, world_circle, fill_color=color)
            self.canvas.create_text(x , y, text=label, font=("Georgia", 14, "bold"), fill="black")
```

```
        outer_circle_spaces = 72
        outer_circle_spaces_1 = 64
        inner_circle_1_spaces = 64
        inner_circle_1_spaces_2 = 56
        inner_circle_2_spaces = 48
        inner_circle_2_spaces_2 = 40
    #human                                  #titan                          #heaven
        self.space_coordinates = {2000: (center_x + 450, center_y - 200), 186: (center_x, center_y),
2500: (center_x + 350, center_y + 320), 2501: (center_x + 450, center_y + 150),
                                  2502: (center_x - 350, center_y - 320), 2503: (center_x - 450, center_y -
150), 2504: (center_x - 350, center_y + 320)}
                                  #animals                       hungry ghost                    hell
        line_coordinates = {}

        for i in range(outer_circle_spaces):
            angle = math.radians(i * (360 / outer_circle_spaces))
            x1 = center_x + outer_circle_radius * math.cos(angle)
            y1 = center_y + outer_circle_radius * math.sin(angle)
            nearest_inner_circle_radius = min(inner_circle_1, buffer_circle_1, inner_circle_2,
key=lambda x: abs(outer_circle_radius - x))
            x2 = center_x + nearest_inner_circle_radius * math.cos(angle)
            y2 = center_y + nearest_inner_circle_radius * math.sin(angle)
            line_coordinates[i] = ((x1 + x2) / 2, (y1 + y2) / 2)

            if i not in [0,9,18,27,36,45,54,63]:
                draw_line(self.canvas, x1, y1, x2, y2, color="black")

            # Display the numbers at the coordinates of the spaces
            space_center_x = (x1 + x2) / 2
            space_center_y = (y1 + y2) / 2


        for i in range(outer_circle_spaces_1):
            angle = math.radians(i * (360 / outer_circle_spaces_1))
            x1 = center_x + outer_circle_radius * math.cos(angle)
            y1 = center_y + outer_circle_radius * math.sin(angle)
            nearest_inner_circle_radius = min(inner_circle_1, buffer_circle_1, inner_circle_2,
key=lambda x: abs(outer_circle_radius - x))
            x2 = center_x + nearest_inner_circle_radius * math.cos(angle)
            y2 = center_y + nearest_inner_circle_radius * math.sin(angle)
            self.space_coordinates[i] = ((x1 + x2) / 2, (y1 + y2) / 2)


            # Display the numbers at the coordinates of the spaces
```

```
            space_center_x = (x1 + x2) / 2
            space_center_y = (y1 + y2) / 2

            if (i + 1) % 8 == 0:
                self.bodhi_spaces.append(i)
                self.canvas.create_text(space_center_x, space_center_y, text=str("bodhi"), fill="blue")
            if (i + 4) % 8 == 0 and (i+1) % 8 != 0:
                self.karma_spaces.append(i)
                self.canvas.create_text(space_center_x, space_center_y, text=str("karma"), fill= "red")
            else:
                self.canvas.create_text(space_center_x, space_center_y, text=str(i + 1))  # Numbering
the space

        for i in range(inner_circle_1_spaces):
            angle = math.radians(i * (360 / inner_circle_1_spaces))
            x1 = center_x + inner_circle_1 * math.cos(angle)
            y1 = center_y + inner_circle_1 * math.sin(angle)
            nearest_inner_circle_radius = min(inner_circle_2, buffer_circle_2, key=lambda x:
abs(inner_circle_1 - x))
            x2 = center_x + nearest_inner_circle_radius * math.cos(angle)
            y2 = center_y + nearest_inner_circle_radius * math.sin(angle)
            line_coordinates[i] = ((x1 + x2) / 2, (y1 + y2) / 2)

            if i not in [0,8,16,24,32,40,48,56]:
                draw_line(self.canvas, x1, y1, x2, y2, color="black")

            # Display the numbers at the coordinates of the spaces

            space_center_x = (x1 + x2) / 2
            space_center_y = (y1 + y2) / 2

        for i in range(inner_circle_1_spaces_2):
            angle = math.radians(i * (360 / inner_circle_1_spaces_2))
            x1 = center_x + inner_circle_1 * math.cos(angle)
            y1 = center_y + inner_circle_1 * math.sin(angle)
            nearest_inner_circle_radius = min(inner_circle_2, buffer_circle_2, key=lambda x:
abs(inner_circle_1 - x))
            x2 = center_x + nearest_inner_circle_radius * math.cos(angle)
            y2 = center_y + nearest_inner_circle_radius * math.sin(angle)
            self.space_coordinates[i + outer_circle_spaces] = ((x1 + x2) / 2, (y1 + y2) / 2)


            # Display the numbers at the coordinates of the spaces

            space_center_x = (x1 + x2) / 2
            space_center_y = (y1 + y2) / 2
```

```
            if (i + 1) % 8 == 0:
                self.bodhi_spaces.append(i)
                self.canvas.create_text(space_center_x, space_center_y, text=str("bodhi"),fill="blue")
            if (i + 4) % 8 == 0 and (i+1) % 8 != 0:
                self.karma_spaces.append(i)
                self.canvas.create_text(space_center_x, space_center_y, text=str("karma"), fill="red")

            else:
                self.canvas.create_text(space_center_x, space_center_y, text=str(i +
outer_circle_spaces + 1))

            # Numbering the space

        for i in range(inner_circle_2_spaces):
            angle = math.radians(i * (360 / inner_circle_2_spaces))
            x1 = center_x + inner_circle_2 * math.cos(angle)
            y1 = center_y + inner_circle_2 * math.sin(angle)
            nearest_inner_circle_radius = min(inner_circle_1, inner_circle_3, key=lambda x:
abs(inner_circle_2 - x))
            x2 = center_x + nearest_inner_circle_radius * math.cos(angle)
            y2 = center_y + nearest_inner_circle_radius * math.sin(angle)
            self.space_coordinates[i + outer_circle_spaces + inner_circle_1_spaces] = ((x1 + x2) / 2,
(y1 + y2) / 2)

            line_coordinates[i] = ((x1 + x2) / 2, (y1 + y2) / 2)

            if i not in [0,6,12,18,24,30,36,42]:
                draw_line(self.canvas, x1, y1, x2, y2, color="black")

            # Display the numbers at the coordinates of the spaces
            space_center_x = (x1 + x2) / 2
            space_center_y = (y1 + y2) / 2


        for i in range(inner_circle_2_spaces_2):
            angle = math.radians(i * (360 / inner_circle_2_spaces_2))
            x1 = center_x + inner_circle_2 * math.cos(angle)
            y1 = center_y + inner_circle_2 * math.sin(angle)
            nearest_inner_circle_radius = min(inner_circle_1, inner_circle_3, key=lambda x:
abs(inner_circle_2 - x))
            x2 = center_x + nearest_inner_circle_radius * math.cos(angle)
            y2 = center_y + nearest_inner_circle_radius * math.sin(angle)
            self.space_coordinates[i + outer_circle_spaces + inner_circle_1_spaces] = ((x1 + x2) / 2,
(y1 + y2) / 2)
```

```
            # Display the numbers at the coordinates of the spaces
            space_center_x = (x1 + x2) / 2
            space_center_y = (y1 + y2) / 2

            if (i + 1) % 8 == 0:
                self.bodhi_spaces.append(i)
                self.canvas.create_text(space_center_x, space_center_y, text=str("bodhi"), fill="blue")
            if (i + 4) % 8 == 0 and (i+1) % 8 != 0:
                self.karma_spaces.append(i)
                self.canvas.create_text(space_center_x, space_center_y, text=str("karma"), fill ="red")
            else:
                self.canvas.create_text(space_center_x, space_center_y, text=str(i +
outer_circle_spaces + inner_circle_1_spaces + 1))  # Numbering the space

        play_button_size = 50
        play_button_x = play_button_size // 2 + 20
        play_button_y = play_button_size // 2 + 20
        draw_play_button(self.canvas, play_button_x, play_button_y, play_button_size,
color="green")

        self.canvas.bind("<Button-1>", lambda event: self.create_pause_menu() if event.x <=
play_button_x + play_button_size // 2 and event.y <= play_button_y + play_button_size // 2 else
None)

        # Add die button
        die_button = tk.Button(self.canvas, text='Roll Die', command=lambda:
self.roll_die(die_button))
        die_button.place(x=20, y=100)  # Adjust the position

        # Draw spokes
        for i in range(8):
            angle = math.radians(i * 45)
            x = center_x + outer_circle_radius * math.cos(angle)
            y = center_y + outer_circle_radius * math.sin(angle)

        # Draw the outer circles
        circle_positions = [
            (center_x + 500, center_y),
            (center_x - 500, center_y),
            (center_x + 410, center_y - 350),
            (center_x - 500, center_y + 200)
        ]
        circle_labels = [ "Bodhisattva", "Karma", "Merit", "Merit"]
        circle_colors = ["wheat", "beige", "powderblue", "thistle"]
```

```
        for pos, label, color in zip(circle_positions, circle_labels, circle_colors):
            draw_circle(self.canvas, pos[0], pos[1], 50, fill_color=color)
            self.canvas.create_text(pos[0], pos[1], text=label, font=("Helvetica", 14), fill="black")

        self.choose_starting_position() #for choosing start

        self.canvas.mainloop()

        self.create_player_piece()


    def create_player_piece(self):
        x, y = self.space_coordinates[self.player.position]
        self.player.piece = self.canvas.create_oval(x - self.player.radius, y - self.player.radius,
                                                    x + self.player.radius, y + self.player.radius,
                                                    fill="blue")

    def launch_about_window():
        about_window = tk.Toplevel()
        about_window.title("About")
        about_window.geometry("300x200")

        about_label = tk.Label(about_window, text="This is a simple game created using Tkinter.")
        about_label.pack(padx=20, pady=30)

        close_button = tk.Button(about_window, text="Close", command=about_window.destroy)
        close_button.pack(pady=10)

        about_window.mainloop()

    def launch_settings_window(self):
        settings_window = tk.Toplevel()
        settings_window.title("Settings")
        settings_window.geometry("300x200")

    def create_buttons(self):
        self.root.title("Game Menu")

        canvas_width = 600
        canvas_height = 425
        self.root.geometry(f"{canvas_width}x{canvas_height}")

        play_button = tk.Button(self.root, text="Play Game", command=self.launch_game,
width=15, height=2, font=("Calibri", 14, "bold"))
        play_button.place(relx=0.5, rely=0.4, anchor=tk.CENTER)
```

```
        about_button = tk.Button(self.root, text="About", command=self.launch_about_window,
width=15, height=2, font=("Calibri", 14, "bold"))
        about_button.place(relx=0.5, rely=0.6, anchor=tk.CENTER)

        settings_button = tk.Button(self.root, text="Settings",
command=self.launch_settings_window, width=15, height=2, font=("Calibri", 14, "bold"))
        settings_button.place(relx=0.5, rely=0.8, anchor=tk.CENTER)


if __name__ == "__main__":
    root = tk.Tk()
    game = Game(root)
    root.mainloop()
```

```
####### cards #########

import pandas as pd
import sqlite3
import tkinter.messagebox as messagebox

def select_bodhi_card():
    # Load the Excel file
    excel_file_path = '/Users/nithi/Documents/GitHub/tae_egame/database/resources/data/Source
Data XLSX/cards.xlsx'
    df = pd.read_excel(excel_file_path)  # Load the sheet into a DataFrame

    # Connect to SQLite database
    conn = sqlite3.connect('TAE_Egame_data.db')

    # Write the DataFrame to SQLite database
    df.to_sql('tae_egame_data', conn, if_exists='replace', index=False)

    # SQL query to select a random Bodhi card context
    query = """
    SELECT card_context, rule_ID, card_contents, card_action
    FROM tae_egame_data
    WHERE card_ID BETWEEN 'card_031' AND 'card_068'
    ORDER BY RANDOM()
    LIMIT 1;
    """

    # Execute the query
    cursor = conn.cursor()
    cursor.execute(query)
    result = cursor.fetchone()

    # Close the connection
    cursor.close()
    conn.close()

    # Print the card context if a card was found
    if result:

        print("Rule ID:", result[1])

        messagebox.showinfo("BODHI CARD", f"BODHI CARD\n {result[0]}\n {result[3]}")

        rule_ID = result[1]

 else:
```

```
        print("No cards found in the specified range.")

def select_karma_card():
    # Load the Excel file
    excel_file_path = '/Users/nithi/Documents/GitHub/tae_egame/database/resources/data/Source
Data XLSX/cards.xlsx'
    df = pd.read_excel(excel_file_path)  # Load the sheet into a DataFrame

    # Connect to SQLite database
    conn = sqlite3.connect('TAE_Egame_data.db')

    # Write the DataFrame to SQLite database
    df.to_sql('tae_egame_data', conn, if_exists='replace', index=False)

    # SQL query to select a random Bodhi card context
    query = """
    SELECT card_context, rule_ID, card_contents, card_action
    FROM tae_egame_data
    WHERE card_ID BETWEEN 'card_001' AND 'card_030'
    ORDER BY RANDOM()
    LIMIT 1;
    """

    # Execute the query
    cursor = conn.cursor()
    cursor.execute(query)
    result = cursor.fetchone()

    # Close the connection
    cursor.close()
    conn.close()

    # Print the card context if a card was found
    if result:
        print("Random Karma Card Context:", result[0])

        messagebox.showinfo("KARMA CARD", f"KARMA CARD\n {result[0]}\n {result[3]}")

    else:
        print("No cards found in the specified range.")
```

Bibliography

Bado-Fralick, N., & Norris, R. S. (2010). Toying with God: The world of religious games and dolls. Waco, TX: Baylor University Press.

Boeker, M., Andel, P., Vach, W., & Frankenschmidt, A. (2013). Game-based e-learning is more effective than a conventional instructional method: A randomized controlled trial with third-year medical students. *PLoS ONE, 8*(12), e82328. https://doi.org/10.1371/journal.pone.0082328

Carvalho, M. B., Bellotti, F., Berta, R., De Gloria, A., Sedano, C. I., Hauge, J. B., ... & Rauterberg, M. (2015). An activity theory-based model for serious games analysis and conceptual design. *Computers & Education,* 87, 166-181

Cosmic Karma [Board game] (2011). Cosmic Karma Game Partners, LLC.

Dankbaar, M. E., Alsma, J., Jansen, E. E., van Merrienboer, J. J., van Saase, J. L., & Schuit, S. C. (2016). An experimental study on the effects of a simulation game on students' clinical cognitive skills and motivation. *Advances in health sciences education : theory and practice*, *21*(3), 505–521. https://doi.org/10.1007/s10459-015-9641-x

Dogen, E. (1999). Shushogi (Training and Enlightenment) [Trans. R.M. Jiyu-Kennett]. In *Zen is Eternal Life* (pp. 94-104). Shasta Abbey Press.

Faulk, S. (2013). *Understanding software requirements*. Portland State University. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Frishkoff, G. A., Collins-Thompson, K., Nam, S., Hodges, L., & Crossley, S. A. (2016). Dynamic Support of Contextual Vocabulary Acquisition for Reading (DSCoVAR): An intelligent tutor for contextual word learning. In *Adaptive educational technologies for literacy instruction* (pp. 69-81). Routledge.

Gentry, S., L'Estrade Ehrstrom, B., Gauthier, A., Alvarez, J., Wortley, D., van Rijswijk, J., Car, J., Lilienthal, A., Tudor Car, L., Nikolaou, C. K., & Zary, N. (2018). Serious Gaming and Gamification interventions for health professional education. *The Cochrane Database of Systematic Reviews*, *2018*(6), CD012209. https://doi.org/10.1002/14651858.CD012209.pub2

Hodhod, R., Cairns, P., & Kudenko, D. (2009). Innovative integrated architecture for educational games: Challenges and merits. *University of York, Computer Science Department*.

Hodhod, R. (2010). *Interactive narrative for adaptive educational games: Architecture and an application to character education* [Doctoral dissertation, Department of Computer Science, United Kingdom].

Ibam, E., Adekunle, T., & Agbonifo, O. (2018). A moral education learning system based on the snakes and ladders game. *EAI Endorsed Transactions on E-Learning*, *5*(17).

IEEE. (2007). *IEEE recommended practice for software requirements specifications* (IEEE Std 830-1998). IEEE Xplore Digital Library. https://ieeexplore.ieee.org/document/720574

IEEE. (2009). *IEEE standard for information technology—Systems design—Software design descriptions* (IEEE Std 1016-2009). IEEE Xplore Digital Library. https://ieeexplore.ieee.org/document/5167255

ISO/IEC/IEEE. (2018). *Systems and software engineering—Life cycle processes—Requirements engineering* (ISO/IEC/IEEE International Standard 29148:2018). IEEE Xplore Digital Library. https://ieeexplore.ieee.org/document/8559686

Jackson, G. T., Boonthum, C., & McNamara, D. S. (2009). iSTART-ME: Situating extended learning within a game-based environment. In *Proceedings of the workshop on intelligent educational games at the 14th annual conference on artificial intelligence in education* (pp. 59-68). Brighton. UK: AIED.

Jackson, G. T., & McNamara, D. S. (2013). Motivation and performance in a game-based intelligent tutoring system. *Journal of Educational Psychology*,105(4), 1036.3

Jiyu-Kennett, P.T.N.H. *Training and Enlightenment: A Buddhist Game and Educational Tool for Adults and Older Children.* 1989. Copyright © 1989. All Rights Reserved. Printed in the United States of America.

Jiyu-Kennett, R. M. (1999). Basic original doctrines essential to Zen. In *Zen is eternal life* (4th ed., pp. 8-13). Shasta Abbey Press.

Jiyu-Kennett, R. M. (1999). The necessity of zazen or meditation practice. In *Zen is eternal life* (4th ed., pp. 23-35). Shasta Abbey Press.

Khantipalo, B. (2013). The Wheel of Birth and Death. In *Access to Insight (BCBS Edition)* November 30, 2013; http://www.accesstoinsight.org/lib/authors/khantipalo/wheel147.html.

Lin, W., Wang, J.-Y., & Yueh, H.-P. (2022). Learning information ethical decision-making with a simulation game. *Frontiers in Psychology, 13*, 933298. https://doi.org/10.3389/fpsyg.2022.933298

MacPhillamy, R. M. D. (2016). The Buddhist precepts. In *Serene reflection meditation* (8th ed., pp. 51–55). Shasta Abbey Press.

McGuire, B. F. (2014). Playing with karma: A Buddhist board game. *Material Religion*, 10(1), 4-29. https://doi.org/10.2752/175183414X13909887177466

McGuire, B. (2022). Permeable boundaries between ritual and play: Religious and ludic possibilities in a Chinese Buddhist board game. *Journal of Chinese Buddhist Studies*, *35*, 109–151. New Taipei: Chung-Hwa Institute of Buddhist Studies.

McKenzie, A. K. (2013). *Conundrum: A serious game informed by Bloom's taxonomy for teaching ethics and social issues* (Master's thesis). University of Saskatchewan, Saskatoon, Saskatchewan.

Morgan, R. M. D. (2016). The Mind of Meditation. In *Serene reflection meditation* (8th ed., pp. 51–55). Shasta Abbey Press.

Ngai, M.-Y. M. (2011). *From entertainment to enlightenment: A study on a cross-cultural religious board game with an emphasis on the Table of Buddha Selection* (Doctoral dissertation, The University of British Columbia).

Rogers, E. (2004). BuddhaWheel™ [Board Game]. Sravasti Arts.

Schlieter, J. (2012). "Simulating Liberation: The Tibetan Buddhist Game 'Ascending the [Spiritual] Levels,'" In *Religions in Play: Games, Rituals and Virtual Worlds*, ed. Philippe Bornet and Maya Burger, Zürich, CH: TVZ Theologischer Verlag Zürich, pp. 93-116.

Schmidt-Madsen, J. (2019). *The game of knowledge: Playing at spiritual liberation in 18th- and 19th-century Western India* (Doctoral dissertation, University of Copenhagen). University of Copenhagen, Department of Cross-Cultural and Regional Studies.

Shute, V. J., & Zapata-Rivera, D. (2012). Adaptive educational systems. *In J. M. Spector, D. Merrill, J. van Merriënboer, & M. Driscoll (Eds.), Handbook of research on educational communications and technology* (pp. 277–294). Taylor & Francis Group. https://doi.org/10.1017/CBO9781139049580.004

Ullman, M. T. (2004). Contributions of memory circuits to language: The declarative/procedural model. Cognition, 92(1-2), 231-270.