CALCULUS QUEST – AN EDUCATIONAL GAME FOR LEARNING CALCULUS

CLUSTER INNOVATION CENTRE

UNIVERSITY OF DELHI

Name of Students

MUDIT GOEL

YASH BIJALWAN

RAVI RANJAN KUMAR

December

2024

MONTH LONG PROJECT SUBMITTED FOR THE PAPER

Single and Multivariable Calculus

CERTIFICATE OF ORIGINALITY

The work embodied in this report entitled **CALCULUS QUEST** has been carried out by **Mudit Goel , Yash Bijalwan and Ravi Ranjan Kumar** for the paper **Single and Multivariable Calculus** I declare that the work and language included in this project report is free from any kind of plagiarism.

(Name and Signature)

Abstract

CALCULUS QUEST – AN EDUCATIONAL GAME FOR LEARNING CALCULUS

Centered by

Mudit Goel

Yash Bijalwan

Ravi Ranjan Kumar,

Cluster Innovation Center, 2024

This project presents *Calculus Treasure Hunt*, an innovative educational game designed to make learning calculus engaging and interactive. Built using Python with the Pygame library, the game combines fundamental calculus concepts with a treasure-hunting gameplay experience. Players navigate a 12x12 grid map to locate treasure chests while answering randomly generated calculus problems involving differentiation and integration. The game adapts dynamically to the player's selected difficulty level—Easy, Medium, or Hard—by varying the complexity of the mathematical expressions and the time allowed for solving problems.

The core features include a visually immersive game interface, real-time feedback on player performance, and a study mode that provides quick access to essential calculus formulas. Players can choose between two gameplay modes: studying calculus through guided examples or attempting to solve timed problems while playing the game. Correct answers are rewarded with points, while incorrect answers or timeouts reduce the player's health (lives). The project integrates symbolic computation with the SymPy library to ensure accurate problem generation and evaluation.

Additionally, the game aims to enhance problem-solving skills by emphasizing quick thinking and reinforcing key concepts through repetition and immediate feedback. By gamifying calculus practice, *Calculus Treasure Hunt* bridges the gap between theoretical learning and practical application, making the subject more approachable for students and enthusiasts alike.

This project showcases the potential of game-based learning in fostering an engaging and enjoyable educational environment for complex subjects like calculus.

I. INTRODUCTION:

Mathematics has long been considered a challenging subject, particularly when it comes to advanced topics such as calculus. Concepts like differentiation and integration often seem abstract, making it difficult for students to grasp their real-world applications. To address this issue, *Calculus Treasure Hunt* was conceived as a gamified learning solution, blending education with entertainment. By leveraging the principles of game-based learning, this project transforms calculus practice into an interactive and engaging experience.

Calculus Treasure Hunt is a Python-based game developed using the Pygame library, designed to reinforce calculus concepts while promoting active participation and critical thinking. The project combines the traditional treasure hunt format with a structured problem-solving approach, where players solve calculus problems to unlock treasures scattered across a virtual map. Each problem is generated dynamically, with varying levels of difficulty to accommodate a wide range of learners.

The gameplay involves navigating a 12x12 grid map, collecting treasure chests, and answering calculus problems under a time constraint. Players can choose between two core quiz types—differentiation and integration—and three difficulty levels: Easy, Medium, and Hard. The game ensures variety and challenge by utilizing the SymPy library to generate calculus problems that range from simple polynomial functions to complex trigonometric, logarithmic, and exponential expressions. Correct answers yield rewards in the form of points, while incorrect answers or timeouts result in penalties such as reduced health (lives).

Beyond the gameplay, the project incorporates a study mode, which serves as a quick reference guide for calculus formulas and rules. This feature allows players to prepare before diving into the challenges of the game, promoting a blend of learning and application.

The project's primary objective is to bridge the gap between theoretical understanding and practical application in calculus. By gamifying the learning process, *Calculus Treasure Hunt* not only makes the subject more approachable but also encourages consistent practice and retention of concepts. This report outlines the technical development, educational significance, and user engagement aspects of the project, demonstrating its potential as a modern educational tool for mathematics.

I.1 Background and Context:

Calculus has been a cornerstone of mathematical sciences for centuries, forming the basis for a wide array of disciplines including physics, engineering, economics, and data science. Its foundational concepts, differentiation and integration, provide tools for understanding change, motion, and accumulation—principles that are critical for solving real-world problems. Despite its significance, calculus remains one of the most challenging subjects for students to master. The abstract nature of its concepts and the steep learning curve often lead to disengagement and anxiety among learners. Despite its essential role in numerous scientific and engineering disciplines, traditional teaching methods frequently fail to fully engage students, relying heavily on rote memorization and static problem-solving exercises.

The rapid advancements in technology and the increasing integration of digital tools in education have opened new avenues for learning, particularly in subjects traditionally considered challenging, such as mathematics. Educational research has highlighted the effectiveness of gamification as a method to increase student engagement and motivation. Gamification involves the incorporation of game-like elements—such as rewards, challenges, and interactivity—into non-game contexts. This approach has been successfully applied in areas such as language learning and early education, yet its application in higher mathematics remains relatively limited. This project, *Calculus Treasure Hunt*, is a response to these challenges, designed to reimagine calculus learning as an interactive and rewarding experience. It is developed to provide an innovative platform for interactive learning, blending advanced calculus concepts with game mechanics to create a more approachable and enjoyable learning experience.

The project is situated at the intersection of three key trends in modern education: gamification, interactive learning, and the integration of technology in pedagogy. Gamification has emerged as a powerful strategy for enhancing learner motivation by incorporating elements of play into educational contexts. The project addresses two key pain points in calculus education. First, it seeks to bridge the gap between theoretical understanding and practical application by presenting calculus problems in a real-time, interactive environment. Second, it tackles the issue of student engagement by embedding mathematical challenges within a treasure hunt game, a format that naturally motivates learners to progress. The game environment is designed to be both visually stimulating and intellectually challenging, ensuring that players remain engaged while reinforcing their calculus skills.

Unlike traditional methods that often rely on repetitive exercises or theoretical lectures, this project leverages the immersive qualities of video games to create a compelling learning environment. The treasure hunt format, combined with dynamically generated calculus problems, encourages players to engage actively with the material. This approach not only reinforces learning through practice but also provides immediate feedback, enabling students to identify and correct their mistakes in real time.

The game is set in a 12x12 grid-based virtual world, where players assume the role of a treasure hunter. Treasure chests scattered across the map are unlocked by solving calculus problems, with each chest offering questions of varying difficulty tailored to the player's selected level—Easy, Medium, or Hard. The problems, generated using the SymPy library, include a mix of differentiation and integration exercises. These problems are presented in varying levels of complexity, ranging from basic polynomials to more advanced expressions involving trigonometric, logarithmic, and exponential functions.

The treasure hunt format is not only entertaining but also strategic. Players navigate a 12x12 grid, collecting treasure chests by solving calculus problems. Each correct answer contributes to the player's score, while incorrect answers or timeouts result in a penalty, such as a reduction in health (lives). This system introduces an element of risk and reward, encouraging players to think critically and manage their time effectively. The grid layout and treasure distribution further add a layer of exploration, requiring players to strategize their movements while solving problems.

The context for developing *Calculus Treasure Hunt* stems from the broader challenges of mathematics education. Existing tools, such as static worksheets or online tutorials, while useful, often fail to capture the interest of students due to their repetitive nature. Similarly, while traditional video games are engaging, they rarely incorporate educational value in a meaningful way. This project seeks to bridge this gap by introducing a game that is both entertaining and educational.

Calculus Treasure Hunt is designed to promote adaptive learning. The ability to select difficulty levels and question types allows learners to tailor their experience to their skill level and areas of interest. This ensures that beginners are not overwhelmed while advanced learners are sufficiently challenged. The integration of a timer in the gameplay further encourages quick thinking and decision-making under pressure, skills that are invaluable in both academic and professional settings.

The educational potential of this project extends beyond individual learning. The game could be employed in classrooms as a supplementary teaching tool, enabling instructors to incorporate interactive activities into their lesson plans. By gamifying calculus, the project not only alleviates the fear and monotony often associated with the subject but also inspires learners to explore its applications in real-world contexts.

In summary, the development of *Calculus Treasure Hunt* is grounded in the need for innovative approaches to teaching complex subjects like calculus. By leveraging the principles of gamification and the capabilities of modern game development, the project offers a fresh perspective on how educational tools can be designed to meet the diverse needs of learners in the digital age.

I.2 Scope and Objectives:

The *Calculus Treasure Hunt* project is designed as a gamified educational tool to enhance the learning experience for students tackling advanced calculus topics such as differentiation and integration. Its scope extends beyond the development of a simple game; it aims to create a platform that bridges the gap between traditional classroom teaching and modern, interactive learning methodologies. This section outlines the comprehensive scope of the project and the specific objectives it seeks to achieve.

Scope of the Project:

The scope of this project encompasses the following dimensions:

1. Educational Gamification

The primary focus of the project is to integrate calculus learning into a gamified environment. By embedding problem-solving tasks into a treasure-hunt-style game, the project addresses the monotony of traditional methods, providing an engaging alternative that motivates students to practice and understand calculus concepts.

2. Dynamic Problem Generation

The game incorporates dynamic question generation using the SymPy library, allowing for a wide variety of differentiation and integration problems to be created on the fly. This ensures a unique experience for each gameplay session, minimizing repetition and keeping the learner challenged.

3. Adaptive Learning Environment

With customizable difficulty levels—Easy, Medium, and Hard—the game caters to learners with varying levels of proficiency in calculus. Each difficulty level adjusts the complexity of the problems, ranging from simple polynomial expressions to more advanced trigonometric, logarithmic, and exponential functions.

4. Study Mode Integration

In addition to the gameplay, the project includes a study mode that serves as a quick reference for essential calculus formulas and rules. This feature enhances the educational aspect of the project, offering players the opportunity to review key concepts in a structured and accessible manner.

5. Interactive Gameplay

The game is set in a 12x12 grid environment, where players navigate to collect treasure chests by solving calculus problems. The design integrates exploration, time management, and strategic decision-making, making the experience both educational and entertaining.

6. Performance Tracking and Feedback

The project incorporates mechanisms to track player performance, including correct and incorrect answers, score, and remaining lives. This feedback system allows players to evaluate their progress and identify areas for improvement, fostering a cycle of continuous learning.

7. Technology Demonstration

By utilizing Python and the Pygame library, the project showcases the potential of open-source technologies for developing educational tools. The integration of symbolic computation through SymPy highlights the capability of modern programming languages to handle complex mathematical operations.

Objectives of the Project:

The key objectives of the *Calculus Treasure Hunt* project are:

1. Enhancing Engagement in Learning

To provide an engaging and interactive platform that transforms calculus from a traditionally intimidating subject into a fun and approachable topic.

2. Improving Conceptual Understanding

To reinforce foundational concepts of differentiation and integration by presenting problems in a practical and application-driven manner.

3. Encouraging Active Learning

To foster active participation by requiring players to solve problems under time constraints, thereby improving their problem-solving speed and accuracy.

4. Personalized Learning Experience

To cater to diverse learning needs by allowing players to select their preferred difficulty level and quiz type, ensuring an inclusive educational environment.

5. Blending Study and Application

To provide a comprehensive learning experience by combining a study mode for theoretical review with gameplay for practical application of calculus concepts.

6. Promoting Technological Literacy

To familiarize students with the integration of technology in education, showcasing how programming and mathematics can work together to create meaningful learning experiences.

7. Measuring Educational Impact

To enable self-assessment through performance tracking, helping learners identify strengths and weaknesses while fostering a growth mindset.

8. Exploring Game-Based Learning for Advanced Mathematics

To demonstrate the feasibility and effectiveness of gamified approaches in teaching higher-level subjects such as calculus, encouraging educators to explore similar methods in other challenging domains.

Relevance of Scope and Objectives:

The project's scope and objectives are aligned with the broader goals of improving mathematical education and leveraging technology to create impactful learning experiences. By focusing on gamification, personalization, and real-time feedback, *Calculus Treasure Hunt* addresses critical pain points in traditional learning methods. It offers a scalable solution for both individual learners and institutions seeking innovative ways to teach complex subjects.

Through its clearly defined scope and ambitious objectives, the project aspires to contribute meaningfully to the evolving landscape of educational technology, setting a precedent for future innovations in gamified learning.

I.3 Achievements:

The *Calculus Treasure Hunt* project was designed with the goal of transforming the way students engage with and learn advanced calculus concepts. By integrating interactive gameplay with educational content, the project has made significant strides in bridging the gap between theoretical learning and practical application. The following are the key achievements of the project:

1. Development of a Gamified Learning Platform

The project successfully created an interactive game that integrates key calculus concepts, such as differentiation and integration, into an engaging treasure hunt format. This gamification of learning has provided a fresh and enjoyable way for students to approach advanced mathematical concepts, moving away from traditional textbook learning.

2. Dynamic Problem Generation and Complexity Management:

A unique feature of the game is its ability to dynamically generate calculus problems in real-time, offering a variety of problems related to differentiation and integration. The use of the SymPy library ensures that each problem is unique, preventing repetitive experiences and consistently challenging the player. The problems range from basic polynomial functions to more complex trigonometric and exponential expressions, maintaining the player's interest and catering to different skill leve

3. Educational Content Integration:

Apart from the gameplay, the project includes a **Study Mode** where players can review important calculus formulas and rules. This addition serves as a valuable educational resource, allowing students to study before tackling the game's challenges. This balance between studying and application ensures that the game is not just fun, but also educationally effective.

4. Immediate Feedback and Performance Tracking

An important accomplishment is the introduction of **real-time feedback**. As players solve calculus problems, the game immediately informs them whether their answers are correct or not, allowing them to learn from their mistakes right away. Additionally, the game tracks the number of correct and incorrect answers, giving players a clear understanding of their progress, and helping them focus on areas where they need improvement.

5. Customizable Difficulty Levels for Personalized Learning:

The project ensures that learners with varying skill levels are accommodated. Players can choose from Easy, Medium, and Hard difficulty levels, which adjust the complexity of the calculus problems and the time allowed to solve them. This personalized approach helps beginners feel more comfortable while offering enough challenge for more advanced learners.

6. Effective Use of Technology:

The project showcases how Python, Pygame, and SymPy can be used to create a functional and enjoyable educational tool. By leveraging these technologies, the project has successfully developed a game that combines real-time symbolic computation with an interactive graphical interface, demonstrating the potential of technology in modern education.

7. Promoting Active Learning and Critical Thinking:

Through the time-limited problem-solving aspect of the game, players are encouraged to think quickly and make decisions under pressure. This element promotes not only active learning but also the development of problem-solving skills, which are critical for mastering advanced topics like calculus. By incorporating key elements of gamification—such as scoring systems, health points, time constraints, and progressive challenges—the project has succeeded in motivating learners.

In conclusion, **Calculus Treasure Hunt** has successfully reimagined how students can engage with and learn calculus by blending educational content with interactive gameplay. Its achievements in dynamic problem generation, personalized learning, real-time feedback, and the gamification of education position it as a pioneering tool for modern mathematics education.

II. FORMULATION OF THE PROBLEM

II.1 PROBLEM STATEMENT

The problem for our report is the struggle of many students to make a basic understanding for the diverse concepts of calculus ranging from limits and derivatives to integrals. This problem is being faced by a lot of beginners as well as some who has some understanding in calculus. It has been need of hour to gamify the experience for those who have even a slight fear for calculus.

It has been seen that traditional methods like books, video lectures and e-learning platforms are being used. They are not only making the concept difficult for the students to understand but also not making them aware of the real-world applications of these concepts. In recent years, there has been some development for gamifying the field of calculus to make it easily available for all the students. But, these games some or the other way lack in their technological output and fun factor for that.

A best solution for this problem is to develop a gamifying platform which can change the attitude of a student towards the concepts of calculus as he would learn instinctively by playing. At that time, he would think to memorize the formulas to win the game and a dynamic game platform which changes based on the development of memorizing power of the students is the need of the hour. Having used this platform, a student will be able to gather some basic knowledge about the calculus concepts. The game will integrate core topics such as limits, derivatives, integrals, and real-world applications, with a particular emphasis on creating an engaging and interactive environment that motivates learners.

The problem is needed to be addressed as there is a growing fear of calculus in the minds of youngsters which is making it boring for them with no more interest left in them to even learn calculus from traditional resources. So, it would help them to recall as well as apply those formulas as well as equations at that very time.

The problem for which a diversifying game is a solution revolves around the following issues: -

1. **Educational Value**: Finding the right balance between the learning objectives and game features can be tough. The content has to align with educational standards, and students may not be familiar with instructional design principles.
2. **Constant discouragement for traditional methods:** - Traditional methods of learning calculus often involve lecture-based teaching, textbook study, and rote problem-solving. While effective for many, these approaches can sometimes be passive or difficult to engage with for some students.
3. **Lack of Active Participation:** - Some students passively absorb content without truly engaging with it or practicing problem-solving on their own. Interactive elements of a game force students to actively solve problems, apply strategies, and make decisions, reinforcing their understanding of calculus concepts.

4. **Sight for continuous practice:** - Mastery of calculus requires continuous practice, but students often lack the motivation or discipline to consistently work through problems. A game can incorporate daily or weekly challenges, encouraging consistent practice through rewards and progression systems. The fun nature of the game makes it easier for students to stay engaged over time.

II.2 Methodology :

The problem descripted above is intended to be solved by using game platform provided by pygame and its related modules. A detailed description of the software and modules is provided as follows: -

1. **PYGAME: -** Pygame is one of the most widely used Python libraries for creating 2D games. It provides easy access to game elements like graphics, sounds, and event handling. Pygame is built on top of the Simple Direct Media Layer (SDL) library, which is a low-level library for handling graphics, audio, and input devices.
   Some of the additional features provided by pygame are: -

- **1.1 Graphics**: Pygame allows you to create 2D games by drawing graphics, displaying images, and handling animations. It supports a variety of formats for images and can handle drawing basic shapes like circles, rectangles, and lines.
- **1.2 Sound**: It supports loading and playing sound effects and music, enabling rich auditory experiences in games.
- **1.3 Event Handling**: Pygame makes it easy to manage user input from various devices like the keyboard, mouse, or even gamepads. It also supports handling system events (e.g., window resizing).
- **1.4 Game Loop**: It provides functionality for creating a game loop that runs continuously, updating the game state and rendering the screen in real time.
- **1.5 Collision Detection:** Pygame includes built-in features for detecting collisions between objects in your game, useful for things like character interactions with walls or other characters.
- **1.6 Cross-platform**: Pygame can be used on multiple operating systems, including Windows, macOS, and Linux, making it versatile for development across different environments.
- 1.7 **Open-source**: Pygame is open-source and free to use, allowing developers to create, share, and modify games and applications.

2. Some of the python modules used are: -
   2.1 PYGAME:- The Pygame module is a collection of Python libraries and modules designed to help create games and multimedia applications. It wraps around lower-level libraries, primarily

SDL (Simple Direct Media Layer), to provide an easy-to-use interface for handling graphics, sound, events, input, and more. Pygame is especially popular for 2D game development due to its simplicity and flexibility.

2.2 SYMPY:- The SymPy module is a powerful Python library for symbolic mathematics. It aims to provide tools for performing algebraic computations and symbolic manipulation, allowing you to perform operations such as solving equations, simplifying expressions, differentiating functions, and much more. SymPy is written entirely in Python and is designed to be easy to use and extend.

2.3 RANDOM:- The random module in Python is part of the standard library and provides functions to generate random numbers, shuffle sequences, and select random items from various data structures. It's commonly used in tasks like simulations, games, statistical sampling, cryptography (though for cryptographic purposes, the `secrets` module is preferred), and any scenario that requires randomization.

2.4 TIME:- The time module in Python is part of the standard library and provides functions for working with time-related tasks, including measuring time intervals, formatting time, and handling delays in program execution. It offers a range of tools to interact with system time, whether you need to record the current time, sleep for a while, or measure the time taken by a function to run. In each mode there is a time limit as follows:

Easy level – 45 seconds

Medium level – 60 seconds

Hard level – 75 seconds

Treasure Map Methodology:

The creation of a treasure map using a matrix of 0s and 1s offers an intuitive and efficient way to represent a game environment. This methodology involves designing the map as a two-dimensional grid, where each element of the matrix corresponds to a specific tile type within the game world. In this system, the value 0 represents a ground tile that the player can traverse, while the value 1 indicates the presence of a treasure chest.

The treasure map is first conceptualized as a simple grid layout. Each row and column in the matrix defines the placement of ground and treasure tiles. For instance, a 12x12 matrix might scatter 1s across various locations to indicate treasures surrounded by traversable 0s. This matrix acts as the foundation of the game, serving as both a visual guide and a logical structure.

The map is then translated into a graphical representation using programming tools. In the game environment, tiles are visually rendered based on their matrix values. Ground tiles, represented by 0s, are drawn using a specified ground image, while treasure tiles, denoted by 1s, are depicted with a treasure chest image. This approach ensures the map is both functional and visually engaging for the player.

Player interaction with the treasure map is a key feature of the design. The player's position within the grid is tracked and updated based on movement inputs such as arrow keys. The game ensures that movement remains within the bounds of the matrix, maintaining a seamless interaction between the player and the map. When the player

moves onto a tile marked as 1, the treasure at that location is collected, and the tile is reset to 0 to indicate it has been cleared.
To enhance the gaming experience, collecting a treasure triggers a calculus question. Players must solve these challenges to earn points, integrating educational content into the gameplay. This adds an interactive and rewarding element to the treasure-collecting process.
Overall, the methodology of using a matrix of 0s and 1s to create a treasure map is both versatile and efficient. It allows for straightforward modifications to the map layout by simply adjusting the matrix values. Additionally, this approach seamlessly integrates the visual, logical, and interactive components of the game, making it an excellent choice for creating dynamic and engaging game environments.

A sample matrix for treasure map game is shown below:

```
# Map Layout (0 = ground, 1 = treasure chest)
game_map = [
   [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0],
   [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0],
   [0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
   [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
   [1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
   [0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0],
   [1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
   [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
   [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1],
   [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
   [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
   [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0]
]
```

Function Generation Methodology:

A list of features is added to the game for having a fun factor as well as to encourage the students to build up a basic knowledge and interest towards the field of calculus. A wide range of questions are being generated by using random module providing various functions ranging from **randint()** to **randrange()**. The questions are characterized by the user to be easy, medium or a tough level category. The questions are generated by the use of a general equation or a statement evident in the code. The code snippet for that task is given as follows: -

The variation of functions in each difficulty level for the calculus game is determined by the complexity of the mathematical expressions generated. Here's a brief explanation:

Easy level: In this level there are questions of simple polynomial functions with 2-3 terms. No trigonometric, logarithmic, or exponential functions questions are included. The functions contain fewer terms and smaller powers, making them straightforward to differentiate or integrate.

e.g. - $x^2 + 3x,\ 2x^3 - 4x^2$ etc.

$$func = sum\left(random.randint(-5{,}5) * x ** iforiinrange\left(1, random.randint(2{,}3)\right)\right)$$

Medium level: In this level there are questions of moderate polynomial functions with 3-4 terms, simple trigonometric functions such as $\sin x$, $\cos x$ etc., logarithmic functions such as $\log(x+1)$, exponential functions such as e^x. This requires handling combinations of basic calculus rules like the product rule or chain rule.
e.g. - $x^3 + 2x + \sin x$, $x^2 + \log(x+1)$ etc.

```
func = sum(random.randint(-5, 5) * x**i for i in range(1, random.randint(2, 4)))
+ random.choice([sp.sin(x), sp.cos(x), sp.log(x + 1), sp.exp(x)])
```

Hard level: In this level there are questions of higher degree polynomial functions questions with 4-5 terms, all trigonometric functions such as $\sin x$, $\cos x$, $\tan x$, $\cot x$, cosec x, sec x, logarithmic functions such as $\log(x+1)$, exponential functions such as e^x. These require comprehensive understanding and application of various differentiation or integration techniques.

e.g. - $3x^4 - 2x^3 + \tan x + \sec x$, $x^5 + \sec x + e^x + \log(x+1)$ *etc.*

```
func = sum(random.randint(-5, 5) * x**i for i in range (1, random.randint(3, 5)))
+ random.choice([sp.sin(x), sp.cos(x), sp.tan(x), sp.csc(x), sp.sec(x),
sp.cot(x), sp.log (x + 1), sp.exp(x)])
```

The game uses the SymPy library to generate these functions dynamically based on the difficulty level and the selected quiz type (differentiation or integration).

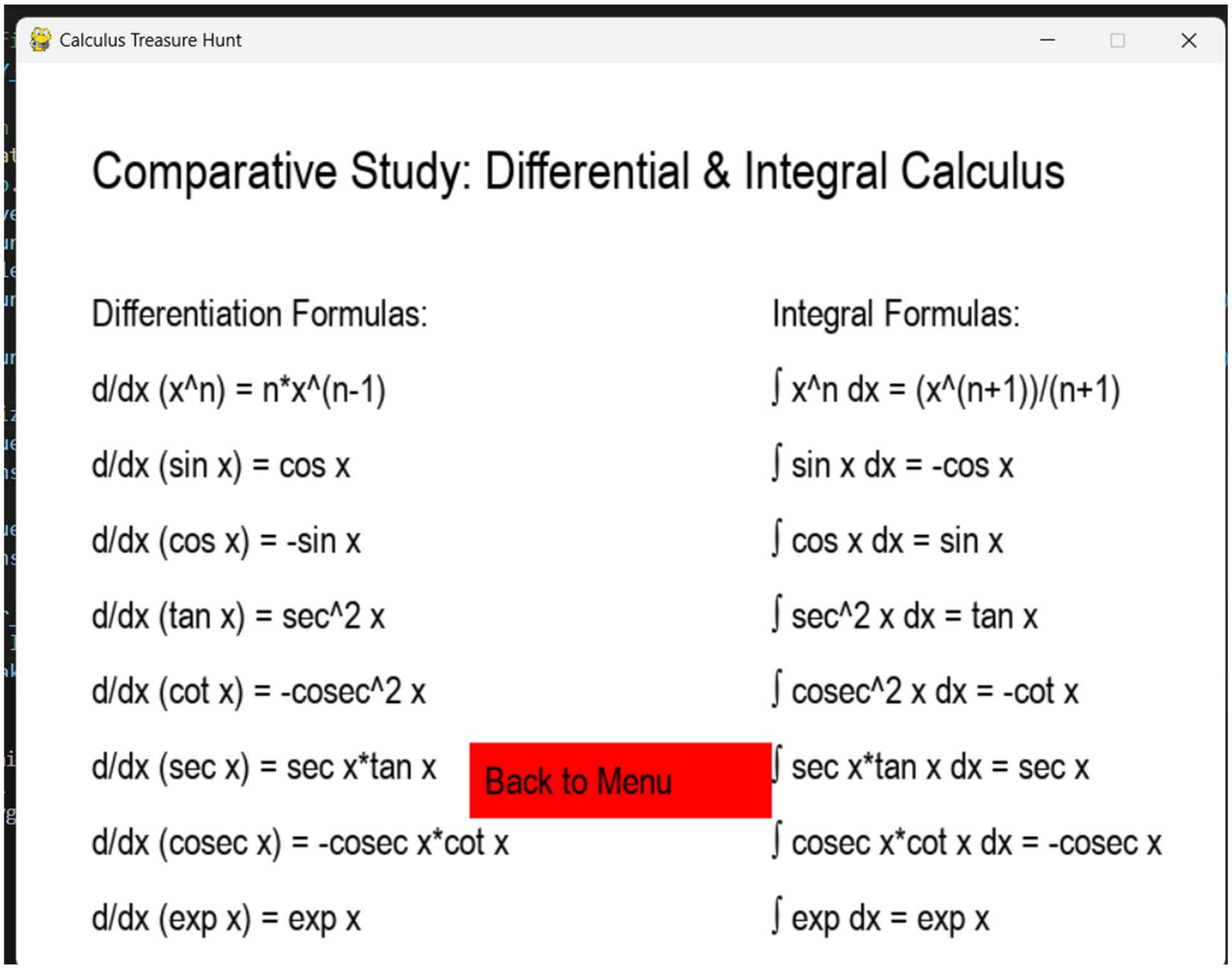
In this way, we have tried to get the best out of the game for those who have an interest in gaming but not in calculus. They can even learn calculus and at the same time can take a lively quiz with health monitoring in the form of lives left which are deducted for every wrong answer and also the interplay of time left indicated by a clock timer works for the betterment in quick response by any player.

II.3 Results :

- ***Execution phase***

STUDY PAGE

The game when started gives the two options to the player to choose from which are either study page which is full of various basic and important formulae and equations admist of which all the game levels revolve around. The study page consists of the basic comparative study of differentiation and integration formulae of those respective functions. A picture showing that study page is as follows: -

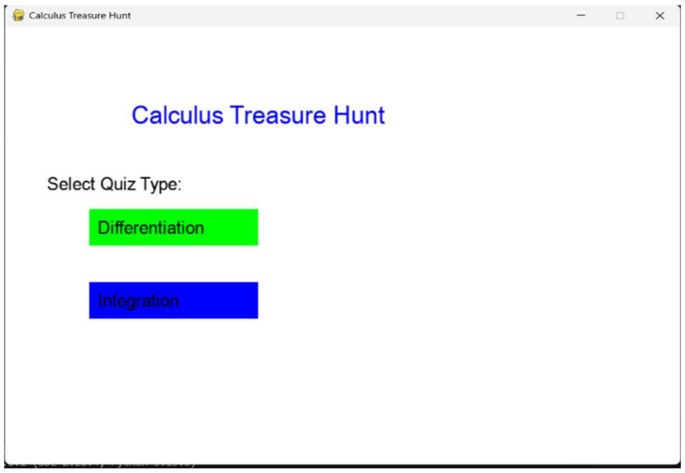


Comparative Study: Differential & Integral Calculus

Differentiation Formulas:	Integral Formulas:
d/dx (x^n) = n*x^(n-1)	∫ x^n dx = (x^(n+1))/(n+1)
d/dx (sin x) = cos x	∫ sin x dx = -cos x
d/dx (cos x) = -sin x	∫ cos x dx = sin x
d/dx (tan x) = sec^2 x	∫ sec^2 x dx = tan x
d/dx (cot x) = -cosec^2 x	∫ cosec^2 x dx = -cot x
d/dx (sec x) = sec x*tan x	∫ sec x*tan x dx = sec x
d/dx (cosec x) = -cosec x*cot x	∫ cosec x*cot x dx = -cosec x
d/dx (exp x) = exp x	∫ exp dx = exp x

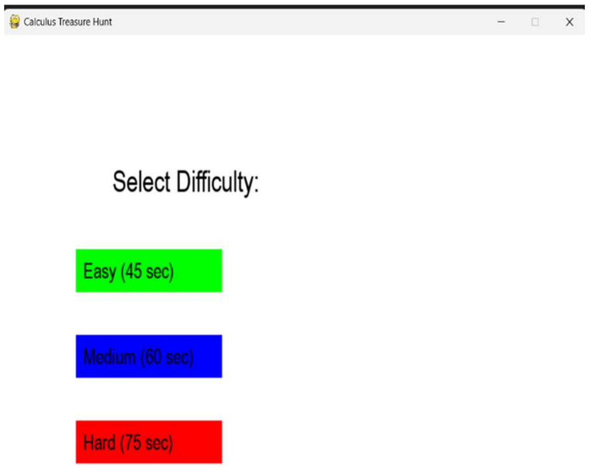
QUIZ TYPE SELECTION PAGE:

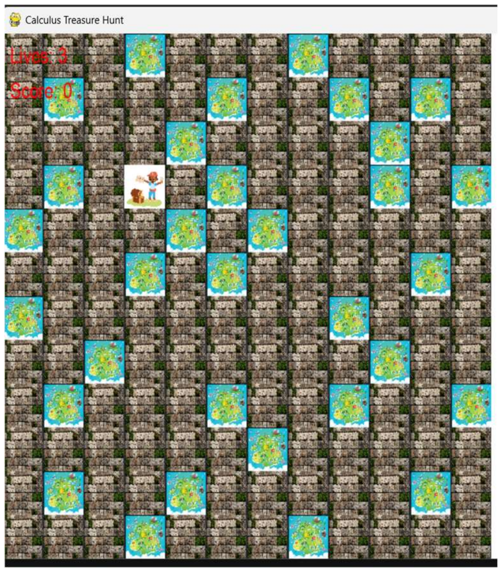
In this page user can select quiz type either differentiation or integration. In each quiz type there are three levels – easy, medium, hard.



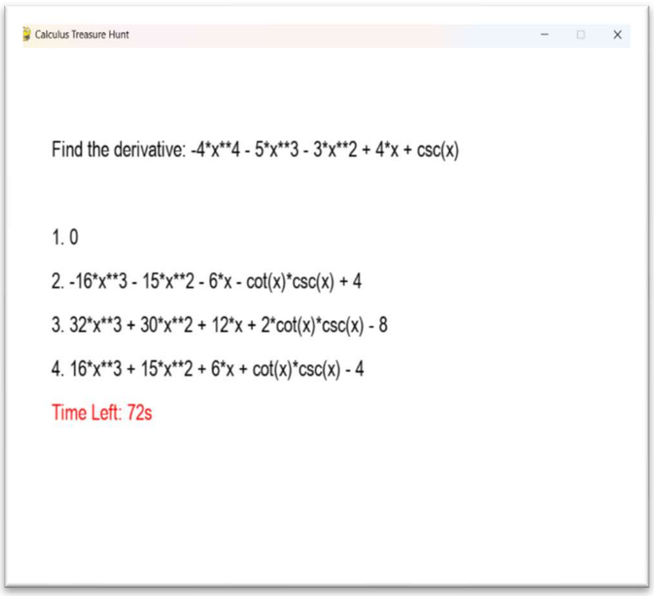
MAIN LEVEL PAGE :

In this page, the user has options to change the level from easy to medium to hard as per the understanding and the need at that time. In this way, the user can switch and play the various levels which differ in the time set provided as well as the difficulty set of questions provided to the user. The treasure map of game is shown here . The total score and no of remaining lives are shown in corner with red color of treasure map page. A sample run conducted is shown here:-





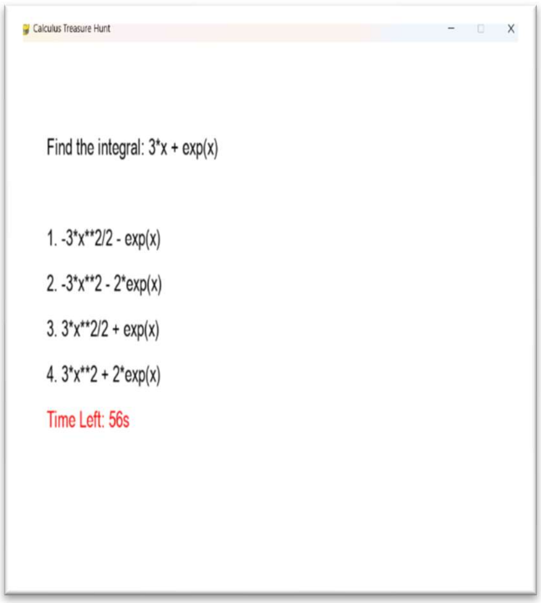
The quiz page is displayed as below in which remaining time (time countdown) according to difficulty level. The user can give answers by pressing num keys 1,2,3 or 4 in the keyboard.


Calculus Treasure Hunt

Find the derivative: -4*x**4 - 5*x**3 - 3*x**2 + 4*x + csc(x)

1. 0
2. -16*x**3 - 15*x**2 - 6*x - cot(x)*csc(x) + 4
3. 32*x**3 + 30*x**2 + 12*x + 2*cot(x)*csc(x) - 8
4. 16*x**3 + 15*x**2 + 6*x + cot(x)*csc(x) - 4

Time Left: 72s

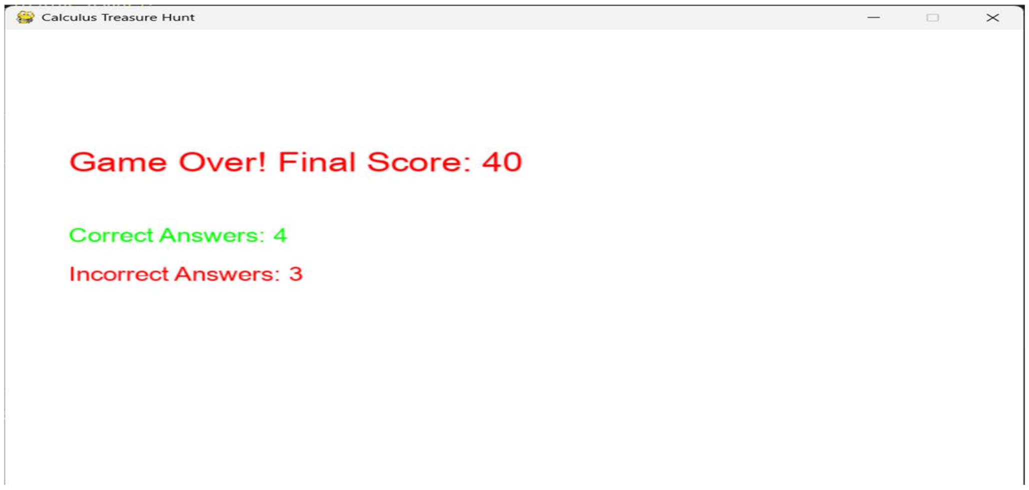

Calculus Treasure Hunt

Find the integral: 3*x + exp(x)

1. -3*x**2/2 - exp(x)
2. -3*x**2 - 2*exp(x)
3. 3*x**2/2 + exp(x)
4. 3*x**2 + 2*exp(x)

Time Left: 56s

The player moves in the treasure through arrow keys of the keyboard. when he hits a block , a question appears on the screen. When the player gives maximum answers correct within the provided three lives, he would get the treasure which is the last view of the game. The player gets 10 points for correct answer, 0 points and loss of one life for incorrect answer. Meanwhile, if the player fails to answer the questions within the stipulated time and the sustained lives, he/she will get the game over prompt and the screen will have a display of their correct and incorrect answers. It is shown as follows:-


Calculus Treasure Hunt

Game Over! Final Score: 40

Correct Answers: 4

Incorrect Answers: 3

This is a view of results we got after executing the game 'Calculus Treasure Hunt' developed for making calculus comparatively easy for them as compared to the traditional methods as well as repeated reading and memorizing the formulae. Also, there would be a real-world application of formulae and a fun factor which will eventually help students to gain interest in calculus.

- User’s Response for Game Creation

User Practical Results

Subject	Percentage
Student	75%
Teacher	72%
Lecturer	60%

CONCLUSION:

Through the project undertaken by us, we have tried to integrate the fun factor of the game ‘calculus treasure hunt’ to the learning of basic concept and formulae of calculus especially Differentiation and Integration which are otherwise difficult to some students. We have tried to make calculus easy and interesting by making the concept of lives and time factor. The player would now try to memorize the concept required to answer the problem within the stipulated time and will have the increase in score and will get the treasure at last.

We have also created a study page for revising the formulae before tackling the game. It will help students try the formulae just after having it recalled by that study page.

Meanwhile, the project has certain shortcomings including the integration of movement in the game by adjusting the coordinates which may be the upcoming level for this treasure hunt. Also, more diversified set of question awaits the game findings. The user will get a basic idea of what the calculus is and how easy it is for those who have the game played. In this way, it is useful for making a good understanding of the calculus with a game mode.

The inclusion of immediate feedback, problem-solving elements, and adaptive difficulty levels ensures that learners can engage with the material at their own pace, enhancing comprehension and retention in a fun and motivating environment. The game provides a unique opportunity to blend entertainment with education, fostering a deeper connection to the subject.

Lastly, a broader range of interactive tools like virtual calculators, graphing tools, or a hint system could make the game even more versatile and useful for students to experiment and explore different problem-solving strategies. With these, we may make a good approach towards making a more effective game with a fun factor and hint for the question.

References :

The references from which the data is taken is: -

[1] DIGITAL GAME BASED LEARNING FOR UNDERGRADUATE CALCULUS EDUCATION:
IGI Global
INTERNATIONAL JOURNAL OF GAMING AND COMPUTER-MEDIATED SIMULATIONS

[2] JOURNAL OF PHYSICS: CONFERENCE SERIES
DEVELOPMENT OF INSTRUCTIONAL MEDIA GAME EDUCATION ON INTEGRAL AND DIFFERENTIAL CALCULUS
https://iopscience.iop.org/article/10.1088/1742-6596/1280/4/042049

[3] HTTPS://SEARCH.APP/?LINK=HTTPS%3A%2F%2FINTERACTY.ME%2FPRODUCTS%2FTREASURE-HUNT&UTM_CAMPAIGN=AGA&UTM_SOURCE=AGSADL1%2CSH%2FX%2FGS%2FM2%2F4

[4] HTTPS://WWW.FUNTRIVIA.COM/TRIVIA-QUIZ/SCITECH/QUEST-FOR-CALCULUS-127041.HTML

REFERENCES FOR IMAGES IN THE GAME:
[1] https://stock.adobe.com/in/608510003
[2 https://pngtree.com/freebackground/seamless-patterns-vintage-game-like-stone-textures-with-ivy-covered-pavement-walls-or-floor-tiles-in-a-bird-s-eye-perspective_15306243.html
[3] https://www.freepik.com/free-vector/hand-drawn-flat-design-treasure-hunt_23908326.html

APPENDIX:

The code used for the game creation is as follows: -

CODE

```python
import pygame
import sympy as sp
import random
import time

# Initialize Pygame
pygame.init()

# Constants
WIDTH, HEIGHT = 800, 600
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
LIGHT_PINK = (255, 182, 193)
FONT = pygame.font.SysFont('Arial', 25)
LARGE_FONT = pygame.font.SysFont('Arial', 35)

TILE_SIZE = 50

# Set up screen
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Calculus Treasure Hunt")

# Game Variables
score = 0
health = 3  # Number of lives
current_question = ""
current_answer = ""
answer_options = []
quiz_type = ""
difficulty = ""
timer = 0
selected_option = -1
correct_answers = 0
incorrect_answers = 0
skipped_questions = 0
question_count = 0
MAX_QUESTIONS = 10
submitted = False

player_pos = [3, 3]  # Starting position
```

```
map_width, map_height = 12, 12

# Load custom images (replace with your own images)
player_image = pygame.image.load('6785156.jpg')  # Your player's image file
player_image = pygame.transform.scale(player_image, (TILE_SIZE, TILE_SIZE))  #
Scale to fit the tile size
ground_image = pygame.image.load('tile.jpg')  # Ground tile image
ground_image = pygame.transform.scale(ground_image, (TILE_SIZE, TILE_SIZE))
chest_image = pygame.image.load('treasure_hunt.png')  # Treasure chest image
chest_image = pygame.transform.scale(chest_image, (TILE_SIZE, TILE_SIZE))
treasure_image = pygame.image.load('treasure.jpg')
treasure_image = pygame.transform.scale(treasure_image, (TILE_SIZE,TILE_SIZE))

# Map Layout (0 = ground, 1 = treasure chest)
game_map = [
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0],
    [0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
    [1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0]
]

# Add difficulty-specific timers
DIFFICULTY_TIMERS = {"Easy": 45, "Medium": 60, "Hard": 75}

# Function to generate calculus questions
def generate_question(quiz_type, level):
    x = sp.symbols('x')
    if level == "Easy":
        func = sum(random.randint(-5, 5) * x**i for i in range(1,
random.randint(2, 3)))
    elif level == "Medium":
        func = sum(random.randint(-5, 5) * x**i for i in range(1,
random.randint(2, 4))) + random.choice([sp.sin(x), sp.cos(x), sp.log(x + 1),
sp.exp(x)])
    else:  # Hard level
```

```
        func = sum(random.randint(-5, 5) * x**i for i in range(1,
random.randint(3, 5))) + random.choice([sp.sin(x), sp.cos(x), sp.tan(x),
sp.csc(x), sp.sec(x), sp.cot(x), sp.log(x + 1), sp.exp(x)])

    if quiz_type == "Differentiation":
        question = f"Find the derivative: {func}"
        answer = sp.simplify(sp.diff(func, x))
    else:  # Indefinite Integration
        question = f"Find the integral: {func}"
        answer = sp.simplify(sp.integrate(func, x))

    answer_options = [answer]
    while len(answer_options) < 4:
        fake_answer = sp.simplify(sp.diff(func, x) if quiz_type ==
"Differentiation" else sp.integrate(func, x)) * random.randint(-2, 2)
        if fake_answer not in answer_options:
            answer_options.append(fake_answer)

    random.shuffle(answer_options)
    return question, str(answer), [str(opt) for opt in answer_options]

# Display text function
def display_text(text, x, y, color=BLACK, font=FONT):
    label = font.render(text, True, color)
    screen.blit(label, (x, y))

# Draw player and treasure chests on map
def draw_map():
    screen.fill(WHITE)
    for y, row in enumerate(game_map):
        for x, tile in enumerate(row):
            screen.blit(ground_image, (x * TILE_SIZE, y * TILE_SIZE))  # Draw
ground
            if tile == 1:
                screen.blit(chest_image, (x * TILE_SIZE, y * TILE_SIZE))  # Draw
treasure chests
    screen.blit(player_image, (player_pos[0] * TILE_SIZE, player_pos[1] *
TILE_SIZE))
    # Display score and health in RED color
    display_text(f"Lives: {health}", 10, 10, RED, FONT)
    display_text(f"Score: {score}", 10, 50, RED, FONT)

# Handle player movement and boundaries
```

```python
def move_player(dx, dy):
    new_x = player_pos[0] + dx
    new_y = player_pos[1] + dy
    if 0 <= new_x < map_width and 0 <= new_y < map_height:  # Check boundaries
        player_pos[0] = new_x
        player_pos[1] = new_y

# Function to display question and options with a timer
def display_question_with_timer(question, options, time_left):
    screen.fill(WHITE)  # Clear screen to display question
    display_text(question, 50, 100, BLACK, FONT)  # Show the question
    y_offset = 200  # Starting point for the options
    for idx, option in enumerate(options):
        display_text(f"{idx+1}. {option}", 50, y_offset, BLACK, FONT)
        y_offset += 50  # Move down for the next option
    display_text(f"Time Left: {time_left}s", 50, 400, RED, FONT)  # Display timer
    pygame.display.flip()  # Update the screen to show changes

# Function to handle player's answer with timing
def handle_player_answer_with_timing():
    global score, health, correct_answers, incorrect_answers
    start_time = time.time()
    while time.time() - start_time < timer:  # Use the timer based on difficulty
        remaining_time = timer - int(time.time() - start_time)
        display_question_with_timer(current_question, answer_options,
remaining_time)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            elif event.type == pygame.KEYDOWN:
                if event.key in [pygame.K_1, pygame.K_2, pygame.K_3, pygame.K_4]:
                    selected_option = int(event.key) - pygame.K_1  # Convert key
to index
                    if answer_options[selected_option] == current_answer:
                        score += 10  # Increment score for a correct answer
                        correct_answers += 1
                    else:
                        health -= 1  # Deduct health for incorrect answer
                        incorrect_answers += 1
                    return  # Exit function after answer
    # If time runs out:
    health -= 1  # Deduct health for timeout
    incorrect_answers += 1

# Main game loop
def main():
    global health, score, current_question, current_answer, answer_options,
quiz_type, difficulty, timer

    # Select study or play option
    study_or_play_page()

    # Select quiz type and difficulty
    start_page()
    difficulty_page()

    # Initialize clock to control frame rate
    clock = pygame.time.Clock()

    # Main game loop
    running = True
    while running:
        # Get events
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()

        # Handle player movement
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
            move_player(-1, 0)
        elif keys[pygame.K_RIGHT]:
            move_player(1, 0)
        elif keys[pygame.K_UP]:
            move_player(0, -1)
        elif keys[pygame.K_DOWN]:
            move_player(0, 1)

        # Check if player is on a treasure chest
        if game_map[player_pos[1]][player_pos[0]] == 1:
            game_map[player_pos[1]][player_pos[0]] = 0  # Remove chest after
opening
            current_question, current_answer, answer_options =
generate_question(quiz_type, difficulty)
```

```python
            handle_player_answer_with_timing()  # Display question and handle
answer with timing

        # Check if health is 0
        if health <= 0:
            game_over()

        # Display score, health, and other stats on the map
        display_text(f"Score: {score}", 10, 10)
        display_text(f"Lives: {health}", 10, 50)

        # Draw the map and player
        draw_map()

        # Update screen
        pygame.display.flip()

        # Limit the frame rate to 30 FPS for smoother gameplay
        clock.tick(30)

# Game over screen
def game_over():
    screen.fill(WHITE)
    display_text(f"Game Over! Final Score: {score}", 50, 150, RED, LARGE_FONT)
    display_text(f"Correct Answers: {correct_answers}", 50, 250, GREEN, FONT)
    display_text(f"Incorrect Answers: {incorrect_answers}", 50, 300, RED, FONT)
    pygame.display.flip()
    pygame.time.wait(5000)
    pygame.quit()
    exit()

# Function to select quiz type
def start_page():
    global quiz_type
    running = True
    while running:
        screen.fill(WHITE)
        display_text("Calculus Treasure Hunt", 150, 100, BLUE, LARGE_FONT)
        display_text("Select Quiz Type:", 50, 200, BLACK)

        diff_button = pygame.Rect(100, 250, 200, 50)
        pygame.draw.rect(screen, GREEN, diff_button)
        display_text("Differentiation", 110, 260)
        int_button = pygame.Rect(100, 350, 200, 50)
        pygame.draw.rect(screen, BLUE, int_button)
        display_text("Integration", 110, 360)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                x, y = pygame.mouse.get_pos()
                if diff_button.collidepoint(x, y):
                    quiz_type = "Differentiation"
                    running = False
                elif int_button.collidepoint(x, y):
                    quiz_type = "Integration"
                    running = False

        pygame.display.flip()

# Function to select difficulty level
def difficulty_page():
    global difficulty, timer
    running = True
    while running:
        screen.fill(WHITE)
        display_text("Select Difficulty:", 150, 150, BLACK, LARGE_FONT)

        easy_button = pygame.Rect(100, 250, 200, 50)
        pygame.draw.rect(screen, GREEN, easy_button)
        display_text("Easy (45 sec)", 110, 260)

        medium_button = pygame.Rect(100, 350, 200, 50)
        pygame.draw.rect(screen, BLUE, medium_button)
        display_text("Medium (60 sec)", 110, 360)

        hard_button = pygame.Rect(100, 450, 200, 50)
        pygame.draw.rect(screen, RED, hard_button)
        display_text("Hard (75 sec)", 110, 460)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                x, y = pygame.mouse.get_pos()
                if easy_button.collidepoint(x, y):
                    difficulty = "Easy"
                    timer = DIFFICULTY_TIMERS[difficulty]
                    running = False
                elif medium_button.collidepoint(x, y):
                    difficulty = "Medium"
                    timer = DIFFICULTY_TIMERS[difficulty]
                    running = False
                elif hard_button.collidepoint(x, y):
                    difficulty = "Hard"
                    timer = DIFFICULTY_TIMERS[difficulty]
                    running = False

        pygame.display.flip()

# Function for study or play option
def study_or_play_page():
    running = True
    while running:
        screen.fill(WHITE)
        display_text("Welcome to Calculus Treasure Hunt", 100, 100, BLUE,
LARGE_FONT)
        display_text("Select an option:", 50, 200, BLACK)

        play_button = pygame.Rect(100, 250, 200, 50)
        pygame.draw.rect(screen, GREEN, play_button)
        display_text("Play Game", 110, 260)

        study_button = pygame.Rect(100, 350, 200, 50)
        pygame.draw.rect(screen, BLUE, study_button)
        display_text("Study Calculus", 110, 360)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                x, y = pygame.mouse.get_pos()
                if play_button.collidepoint(x, y):
                    running = False
                elif study_button.collidepoint(x, y):
```

```python
                    study_calculus_page()
                    running = False

        pygame.display.flip()

# Study page to show formulas
def study_calculus_page():
    running = True
    while running:
        screen.fill(WHITE)
        display_text("Comparative Study: Differential & Integral Calculus", 50,
50, BLACK, LARGE_FONT)
        display_text("Differentiation Formulas:", 50, 150, BLACK, FONT)
        display_text("d/dx (x^n) = n*x^(n-1)", 50, 200, BLACK, FONT)
        display_text("d/dx (sin x) = cos x", 50, 250, BLACK, FONT)
        display_text("d/dx (cos x) = -sin x", 50, 300, BLACK, FONT)
        display_text("d/dx (tan x) = sec^2 x", 50, 350, BLACK, FONT)
        display_text("d/dx (cot x) = -cosec^2 x",50, 400,BLACK,FONT)
        display_text("d/dx (sec x) = sec x*tan x",50, 450,BLACK,FONT)
        display_text("d/dx (cosec x) = -cosec x*cot x",50, 500,BLACK,FONT)
        display_text("d/dx (exp x) = exp x",50, 550,BLACK,FONT)

        display_text("Integral Formulas:", 500, 150, BLACK, FONT)
        display_text("∫ x^n dx = (x^(n+1))/(n+1)", 500, 200, BLACK, FONT)
        display_text("∫ sin x dx = -cos x", 500, 250, BLACK, FONT)
        display_text("∫ cos x dx = sin x", 500, 300, BLACK, FONT)
        display_text("∫ sec^2 x dx = tan x", 500, 350, BLACK, FONT)
        display_text("∫ cosec^2 x dx = -cot x", 500, 400, BLACK, FONT)
        display_text("∫ sec x*tan x dx = sec x", 500, 450, BLACK, FONT)
        display_text("∫ cosec x*cot x dx = -cosec x", 500, 500, BLACK, FONT)
        display_text("∫ exp dx = exp x", 500, 550, BLACK, FONT)

        back_button = pygame.Rect(300, 450, 200, 50)
        pygame.draw.rect(screen, RED, back_button)
        display_text("Back to Menu", 310, 460)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                x, y = pygame.mouse.get_pos()
                if back_button.collidepoint(x, y):
                    return  # Go back to the main menu
        pygame.display.flip()

# Run the game
if __name__ == "__main__":
    main()
```