

|  |  |
| --- | --- |
| Programming Project  Database Management System for Silverdale School. | Abstract  Bank robbing simulator to learn times table  James Surur (Student Silverdale School)  H446 OCR Computing |

Table of Contents

[Analysis 2](#_Toc486501090)

[Justification 2](#_Toc486501091)

[Stakeholders 2](#_Toc486501092)

[Research 2](#_Toc486501093)

[Essential Features 3](#_Toc486501094)

[Limitations 3](#_Toc486501095)

[Requirements 3](#_Toc486501096)

[Success Criteria 3](#_Toc486501097)

[Design 4](#_Toc486501098)

[Decomposing the problem 4](#_Toc486501099)

[Structure of the solution 4](#_Toc486501100)

[Algorithms 4](#_Toc486501101)

[Usability 5](#_Toc486501102)

[Variables 5](#_Toc486501103)

[Testing 6](#_Toc486501104)

[Post Development 7](#_Toc486501105)

[Development 8](#_Toc486501106)

[Evidence 8](#_Toc486501107)

[Prototypes 8](#_Toc486501108)

[Review 8](#_Toc486501109)

[Testing (Development) 9](#_Toc486501110)

[Test Plan 9](#_Toc486501111)

[Failures 9](#_Toc486501112)

[Evaluation (Testing) 10](#_Toc486501113)

[Evidence 10](#_Toc486501114)

[Usability Testing 10](#_Toc486501115)

[Evaluation 11](#_Toc486501116)

[Success Criteria 11](#_Toc486501117)

[Usability 11](#_Toc486501118)

[Successes 11](#_Toc486501119)

[Failures 11](#_Toc486501120)

[Limitations 11](#_Toc486501121)

[Alterations 11](#_Toc486501122)

[Developments 11](#_Toc486501123)

# Analysis

## Justification

The game will be able to help with building an understanding on how to quickly solve maths problems mentally under a timed conditions aiding students in test increasing their ability to problem solve using multiple variables.

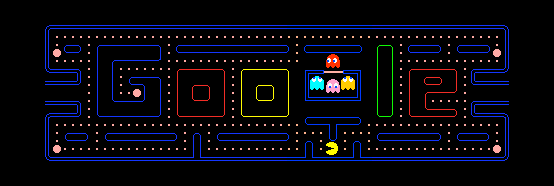
With a target audience of 13-14 students to help increase their mathematical aptitude.

## Stakeholders

Schools can be stakeholders due to the ability of implementing the game as an aide to students learning mathematics as a fun activity to increase engagement within lessons.

Students struggling with visualizing mathematical operations can use the program to better understand how to solve problems using visual representation on how it is ordered.

## Research



Pacman is a third-person action maze with the objective of eating all the pellets.

Pacman use of ghosts as enemies to cause a limited time of when the player can successfully eat the pellets.

Power-ups are spread through the game to act as power ups to help a player strategize on how to beat the ghosts.

Features Implemented

* Series of enemies with movement patterns towards the player
* Map with limited paths to move through.
* Objectives spread through the game.
* Third person view

A video game with a cartoon character

Description automatically generated

Is a game where you answer questions then are able to play the game choosing from a range of types of maths problem. The game uses power ups and the ability to lay eggs to make quick decisions to avoid hitting the ground due to the player character automatically moving forward.

Features Implemented

* User interface for types of questions
* Showing what the right answer would be.
* Questions are needed to be answered correctly to progress.

## Essential Features

* Player movement through standard controls to make a more interactive game as it is an action maze rpg.
* Enemies with dynamic paths respective to the user’s movement so it allows for more variations between games instead of a causing a path that can be easily found and avoided, so enemies are able to react to a player’s keyboard input.
* Database of maths problems that can be solved by 13–14-year-olds, allowing the development of skills for the intended as making questions too complex would make questions unsolvable in the intended timeframe and won’t build on their maths skills.
* Collison with map to limit movement of the player character preventing the player from being able to move in a straight line to complete objectives, as otherwise would prevent the games’ purpose of being and an action maze rpg and closer to a simple memory game.
* Third person top-down view to allow the player to strategize about which objectives should be completed first.
* Maze like map
* Enemies that follow the player and can cause a game over when the sprite of the plater and enemies collide.

## Limitations

Adding a multiple player to the game won’t be feasible due to unavailability of servers to be able to provide a non-local connection to be able to successfully run a multiplayer.

Detailed ai would be limited to how dynamic the pathing due to a limit of programming knowledge and development time in creating a large-scale decision tree for an enemy’s movement.

Procedural generation of map would be unable to be done., due to requiring too much processor power, and the time needed due to the map needing to be broken down into segments resulting in .

## Requirements

Keyboard and a computer

Python- to allow easy implementation of animation and linking of scripts with different objects in the program.

## Success Criteria

* Implementation of player movement through a 2d map using wasd allowing the player to move up and down as well as left and right.
* Pathing for enemies where they move following a generic path towards the coordinates of the player e.g. Up and down until reaching the player.
* Collision between characters and the map so the player can’t move where walls are present in the labyrinth as well as enemies when following the player will only be able to go where there aren’t any walls when chasing the player.
* Objective pause the game stopping enemies in place until the question is answered.
* User is able to choose what type of questions are asked through a button in the menu at the start.
* Objectives are removed after the player answers a question correctly, visually disappearing with program treating it as not present.
* When player collides with an objective screen change to display question.
* When start is pressed display a timer of 240 seconds counting down and lowering time available for questions that are answered incorrectly by 25 seconds.
* When start is pressed display a map with a maze with enemies generated in empty rooms far away not in line with the walls of the maze and the player starts at the entrance of the maze.

# Design

## Decomposing the problem

## Structure of the solution

The implementation of the player and movement would be the first step of completing the program due to the nature of the game as an action labyrinth player movement is a key aspect of the game that would allow enemies to respond to in a dynamic way increasing the engagement with the game the user would have.

Next enemy pathing would be introduced to respond to user inputs and the direction of the player character to allow for greater user control and allow strategies for the user to manipulate enemy pathing to successfully complete objectives.

Map for the user to navigate with walls to limit movement and create pressure. The map is an essential feature for the genre of the game and for the game to function properly the map would be required to have a significant time of development.

Objectives that proc questions, mathematical questions are needed to achieve the aims of the program and help students improve their maths skills therefore after having the game in a working condition, questions should be implemented.

Menu should be created afterwards to allow users to choose from selection of questions in an area they would like to improve in.

## Algorithms

Player movement

Procedure move(key\_input):

If key\_instance\_is\_pressed(W) Then

Player.y -=1

Elif key\_instance\_is\_pressed(S) Then

Player.y+=1

Elif key\_instance\_is\_pressed (D) Then

Player.x+=1

Elif key\_instance\_is\_pressed (A) Then

Player.x -=1

End procedure

Procedure draw\_player()

Set\_position(player.x, player.y)

Draw\_sprite(player\_sprite)

A diagram of a computer program

Description automatically generated with medium confidence

## Usability

## Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | reason | validation |
| Question | String | Stores the question for the player to answer | Changes from the input of the questions chosen by the user |
| Enemy | class | To allow generation of multiple instances that have similar behaviour and properties | Creates instances when game is pressed play |
| objectives completed | Boolean | Once true allow player to access vault to complete game | Set to False until completed objectives sets it to true |
| Timer | integer | Shows how much time the player has before game over screen | Once a second passes decrease time by 1 from a total of 320 |
| completed objectives | integer | Allows the ability to set all\_objectives\_completed to be true | Once user answers a question successfully increase count by 1 |
| Multiplication | Boolean | Allows user to choose what type of question to asked | Button that allows the user to click to choose value to be added. |
| Start | Boolean | Starts the game and spawns in map and enemies | Allow user to only click on the start button to allow enemy and the player instance to spawn |
| Answer | Integer | Allows user to answer the question asked. | Only allows integer values and uses data sanitisation if a float is inputted |
| Player\_caught | Boolean | If player hurt box collides with an enemy causes the game to end | Hurt box of assets match the visual representation seen by the player. |
| Map collision | Boolean | Prevents user from moving through walls of the map preventing no clipping. | False until collides with map border the true |

## 

## Testing

Testing would be done using boundary and erroneous data to see if questions react logically to incorrect or incorrect inputs. Furthermore, issues that may arise is memory loss to whether an objective is completed with the objectives still appearing on the map. So, upon completion of an objective testing should occur to see if the game is able to delete that instance of the objective. Player may not have proper collision so moving the player around the map to check whether areas to where the user should be able to move through are blocked.

## Prototypes

A black and white maze

Description automatically generated

A cartoon of a phone

Description automatically generated

## Post Development

# Development

## Evidence

## 

The implementation was not able to be done due to requiring the player asset to be made first. Allowing the player to be loaded in.

## A pixelated cartoon of a person running Description automatically generated

Player Character

A black background with white text

Description automatically generated

A computer screen shot of a code

Description automatically generated

A pixelated video game of a person running

Description automatically generated

Code has error where the player is not able to move and is stuck on a background with pygame window not responding.

## 

After some research error was found that code needed to when to end loop which otherwise caused slowdown within the code.

Player is able to move but background adds player sprite colours when moving. Player also can go beyond border of map and become stuck.

## A computer code with text Description automatically generated

Code creates background for player to be able to move in.

A screenshot of a computer program

Description automatically generated

Code turned into a class to export into a main file more user friendly and concise.

A screenshot of a computer code

Description automatically generated

Code does not load the player and the background on the same pygame window, loading separately with the window containing the map being unable to be closed.

## A screen shot of a computer program Description automatically generated

To initialize the player in the same pygame screen called player class to world where a sprite group is created during the creation of Map class to allow the class to successfully draw the player character on the screen.

Player is too large for the area of the map so using pygame.transform to scale the image to a quarter of the size to fit within the map boundary.

## Review

# Testing (Development)

## Test Plan

## Failures

# Evaluation (Testing)

## Evidence

## Usability Testing

# Evaluation

## Success Criteria

## Usability

## Successes

## Failures

## Limitations

## Alterations

## Developments