OCR GCE A

COMPUTER SCIENCE PROJECT

H446-03

Name : Alexander Mills

Candidate Number : <INSERT CANDIDATE NUMBER>

<Institution Name> : <INSERT CENTRE NUMBER>

Title of Project : <INSERT PROJECT TITLE>

H446-03 – Project CONTENTS

Table of Contents

[A. Analysis 3](#_Toc88039851)

[Outline 3](#_Toc88039852)

[Stakeholders 3](#_Toc88039853)

[Game Research: Tetris 4](#_Toc88039854)

[Game research: Hue 7](#_Toc88039855)

[Survey 8](#_Toc88039856)

[Interview 8](#_Toc88039857)

[Limitations and Scope 8](#_Toc88039858)

[Why this Solution is Suited to a Computation Solution 9](#_Toc88039859)

[Abstraction 9](#_Toc88039860)

[Thinking Ahead 9](#_Toc88039861)

[Thinking Procedurally 9](#_Toc88039862)

[Thinking Logically 9](#_Toc88039863)

[Thinking Concurrently 9](#_Toc88039864)

[Proposed Feature List 9](#_Toc88039865)

[Success Criteria 9](#_Toc88039866)

[B. Design 9](#_Toc88039867)

[Systems diagram 9](#_Toc88039868)

[C. Developing the coded solution (“The development story”) 10](#_Toc88039869)

[D. Evaluation 10](#_Toc88039870)

[Project Appendixes 11](#_Toc88039871)

# A. Analysis

## Outline

There is great academic pressure on students to perform to the best of their ability. To achieve this, students must study for longer, increasing stress levels and generating concern about whether time is being used effectively. There is a subsequent reduction in time spent on activities that don’t tangibly benefit academic performance like gaming and other recreation. This has an adverse effect on mental health as it sets up a poor work life-balance and means there is no opportunity to de-stress, creating an unstainable feedback loop which will hinder long term attainment.

To rectify this, I shall develop a game which 2d top-down tile game that heavily focusses on puzzle solving and systematic thinking. This will allow students to practice their problem solving and logical reasoning skills in a relaxed, enjoyable, and interactive game environment. This allows them to decompress, improving work-life balance due to a more sustainable method of practicing cognitive skills than studying. To successfully develop this solution, I will draw inspiration from other puzzle solving games such as Retro classics like Tetris(1984) and more modern examples like Portal(2007) and Hue(2016). This will allow me to evaluate existing solutions within this genre and which features are needed to ensure the game holds up to the stakeholders’ expectations and meets their needs.

## Stakeholders

The target demographic of the game will be students in the age range of 15 to 18 who enjoy regular problem solving and logical thinking. This demographic covers a wide range of abilities; therefore, the game must have an array of tiered difficulty levels to ease beginners into the game while allowing advanced players to still enjoy it.

It is designed to be played after a study session to unwind, so the user will likely have a computer available, on which they play the game. This means the game doesn’t need to be portable, so will be controlled by mouse and keyboard. As the game will be used to unwind and relax, it will have a simple, easy to understand control scheme; this will make it easier to learn and less taxing to use. To ensure that it is accessible to as many as possible, there will be very minimal text, having a symbol focused UI to overcome language barriers. The colour pallet of the game will use colours which are not too bright and have minimal blue; this will ensure it is pleasant on the eyes and not alarming, allowing the user to relax.

I have selected Benjamin Dodwell and Mate Fehevari to represent the target demographic. They are both 17 year old students who play videogames regularly. Their experience with similar games will allow them to give clear and well-judged feedback on my game, and how it compares to similar ones in the industry, allowing me to ensure my game meets the target demographics’ needs effectively. They are also close contacts, so I will be able to regularly receive incremental feedback throughout the development process.

## Game Research: Tetris

Tetris is a 2d puzzle game where the player stacks blocks on a 10x20 grid. The square blocks come in groups of 4 called “tetrominos”, which can have many different shapes. They fall to the bottom of the board, and then stop falling, landing on top of any blocks that had previously fell. Should a full row be completed when the falling blocks are placed, this row is cleared, scoring the player some points. This makes for an engaging game where the player must organise a random stream of shapes into a compact pile at the bottom of the board, figuring out which shapes fit where to keep the board organised.

The game starts slowly, with the blocks falling slower. This allows inexperienced players to get used to the game mechanics . As more rows are cleared and more points are scored, the pieces fall faster, allowing the player less time to decide where to place the piece. This makes the game much more stressful and difficult for all but the most experienced players as even a small error can cause big problems, causing the blocks to pile up towards the top of the board, at which point the game is over.

To incentivise more advanced strategies, the game rewards clearing multiple lines at once, rewarding the user with more points. If they clear 4 lines in one go (the maximum possible), they score 8 times as many points as a single line. This leads to players risking building up larger piles so that they can clear more rows at once, earning more points more quickly.

Main menu: Graphical user interface

Description automatically generated

The game’s main menu is the first thing that a potential player sees, therefore it is designed to introduce the players to the game, setting the colour scheme, theme, and branding. To help new players learn the game, there is a question mark button, which shows the controls, how to play the game and the language used to describe gameplay. My menu should contain all these features to make it usable an engaging.

The same UI “windows” are used in both the menu and the actual game. Hence the start menu has features that are blanked out, such as the “NEXT” and “HOLD” queues, which could be distracting or confusing for a new user. It also makes the UI over-crowed, so I will in my game I won’t be re-using UI elements to reduce clutter.

Gameplay:

A picture containing text, parking

Description automatically generated

The main game screen reuses the elements of the menu, so is familiar, though now all the elements are used. The bright colours on a dark background makes the game easier to look at, as well as distinguishing the individual sprites in the game and drawing the user’s attention to the important features. The indicator of where the blocks will fall makes it easier for the user to see what the game will do next(where the block will land), reducing the chance of the user placing a block in the wrong place – this makes the game less annoying and therefore more enjoyable for the user; my game must also focus on this to meet the user’s needs.

Pause Menu:

Graphical user interface

Description automatically generated

The pause menu allows the user to stop the game and return to it later. This makes the game more convenient to play as the user can pick it up and put it down as they want. This will be less important in my game as each level will be played all in one session, though it will still be needed. The menu also offers a tutorial section for teaching inexperienced users and an options menu to allow the user to configure the game to their play style. My game should also have ample configurability to allow the user to have a comfortable gaming experience.

## Game research: Hue

Hue is a puzzle-based side scrolling adventure game with the goal of exploring the map and progressing the story line. The core game mechanic is that the player can change the colour of the background, making game objects of the same colour disappear, allowing the player to pass through them. With multiple colours, the puzzles become very intricate, requiring the player to carefully develop a strategy to deal with each new level, skilfully timing the switch between colours to avoid coloured hazards, move game objects around each other and traverse the coloured platforms to the exit. This mechanic makes for a more enjoyable and rewarding experience for the user as they must reason through how to make every move, and therefore I will implement a similar system for my game.

The game also makes strong use of a storyline developed by both narration and dialogue boxes from NPCs. The narration is triggered by the player finding letters, which are placed in longer, labyrinth style levels which are less challenging, allowing the player to absorb the story. The storyline adds depth and reason to the game, giving the player a reason to progress to the next area to further understand the situation. This makes for a more immersive and engaging gaming experience, though a good story takes time to be written and will need narration, meaning this is out of the scope of my game.

Typical level:

Graphical user interface, diagram, schematic

Description automatically generated

The colour scheme of the game is very focused around the 8 colours of the colour wheel, so they are a repeating theme throughout the whole game. The key game objects are in bright colours, which is both for the functionality and to highlight them to the player. The monochrome background complements the colours and is easy on the eyes, making it easier for the player to look at as it makes no use of bright or startling colours. I will make use of a similar colour scheme for my game, as it will make my game more relaxing to play, while still having visual interest.

The level design makes use of hazards, which the player must avoid by making use of the colour changing mechanic. These force the player to carefully time their inputs, making the game more challenging. The level also has multiple objectives: the player must acquire a key first before passing through the exit This again facilitates more advanced puzzles. To make my puzzle game equally fun, I should incorporate all these level design queues. Each level has been manually designed, making them detailed, though I don’t have time to design levels to this degree, so mine will have to be procedurally generated.

Pause Menu:

A picture containing timeline

Description automatically generated

The pause menu allows the user to pause the game, allowing them to return to it later. It also provides some configuration menus for the user to tailor their experience to their needs. This includes a controls menu, where the user can learn the controls or configure them, a video menu where the user can configure the display resolution and full screen. It also has a colour-blind accessibility option, which is important as being able to distinguish colours is critical to the game, ensuring the game can be played by all potential stakeholders. The audio menu allows the user to control the volumes of different aspects of the game to their liking. These are all quality-of-life features, which enhance the rest of the user experience, and therefore will need to be a part of my game if it is to be enjoyable to play.

## Game Research: World’s Hardest Game

World’s hardest game is a puzzle game where the player must navigate through mazes to the exit, collecting objectives before exiting. The mazes are 2d and are viewed from top down, so the player can immediately see all parts of the maze. This means that the player can heavily strategise how they are going to proceed through the level, but there is nothing to explore.

The core mechanic that makes the game much harder is the hazards moving about the maze. If the player touches one, they instantly die and return to the nearest checkpoint. They all follow pre-defined paths around the level but most move very quickly. The levels are designed such that all places in the maze baring a few have hazards moving over them, meaning the player must keep moving to stay alive, and as they are so close together, the player must perfectly time their inputs to move between them without hitting them, making the game very difficult. While this makes the game fun, it is also very stressful, something I want to avoid, so in my game there will be vastly fewer hazards and if they move, they will be much slower.

Typical level:

Diagram, schematic

Description automatically generated

The levels are all manually designed and have a standard structure: the checkpoints are green areas, the hazards are blue circles, objectives are yellow circles, and the player is a red square. This means the player knows exactly what they are doing each level, making the game intuitive to play. The maze has a checkerboard floor which clearly shows the game is tile based, allowing the player to judge the position and motion of the hazards. Manual layout makes for some clever and challenging level designs, though time must be invested to compose all the levels. As my game will need many levels, it will have to be procedural, but this will work well as it can generate a standardised colour scheme.

## Survey

### Planning

To gauge the needs of a larger group of potential stakeholders, I will use a survey to collect their opinion on how features of the gamer will be designed. This will allow me to make informed decisions about how the game should look and feel to play.

|  |  |  |
| --- | --- | --- |
| Question | Input type | Function |
| How important are graphics to make a puzzle game enjoyable? | Slider: 1 to 10  Comments box | Gauges how much work must be put into graphics to meet user needs |
| How much control over graphics is needed in the settings? | Multi choice:   * No options * basic options: resolutions, vsync, Fullscreen * advanced: frame rate, rendering settings, toggleable visuals * extensive: full colour scheme configurability, all rendering settings | Allows me to develop a suitable graphics menu to make the game accessible for all users |
| How important are visual effects and animations to make a puzzle game enjoyable? | Slider 1 to 10  Comments box | Gauges how much work needs to be put into visual effects and animations |
| How important are Sound effects to make a puzzle game enjoyable? | Slider 1 to 10  Comments box | Gauges how much work needs to be put into the game’s sound design |
| How much control over sound is needed in the settings? | Multi choice:   * no options * a slider for game volume, and a slider for music volume * all game sounds have individual siders | Allows me to design suitable sound settings that will allow users to configure their game to their interests |
| How important is Background music to make a puzzle game enjoyable? | Slider 1 to 10  Comments box | Gauges how important background music is for the users to enjoy the game |
| How much time would you want to spend per level when playing a puzzle game? | Numerical input in minutes  Comments box | Allows me to tune the level length so the game can be challenging for users but not enduring |
| How many times would you want to restart a level before completing it? | Numerical input  Comments box | Allows me to adjust how many hazards there are in a level |
| Should the levels contain checkpoints? | Boolean  Comments box | Determines if users want checkpoints or not, and thus determines if I will implement them |
| How should the game be titled? | Multi choice:   * based on visual theme * based on the style of puzzles * based on a narrative | Ensures that the title of the game conveys the theme and style of game to potential players well |
| Are there any other features which you would like to see in a puzzle game? | Comments box | Allows any other responses from the users, so they can input any other features they would like to see in the game |

### Survey Response review

Graphics:

Chart, bar chart

Description automatically generated

From the graphics part of the survey, it is evident that potential users prefer graphical fidelity over visual effects., though they are both very important This means that I will have to spend more time on textures and sprites, ensuring they are high resolution with ample colour depth. I won’t have time in this project to make them to the level required, so I will have to find some copyright free asset packs that work well together. These asset packs should also come with animations, allowing me to add some visual effects to the game quickly, though that isn’t as important to the overall quality.

The users only need a simple settings menu which offers basic configuration for the game graphics, so I will implement a single graphics menu screen with configurable resolution and Fullscreen options.

Chart, bar chart

Description automatically generated

Chart, pie chart

Description automatically generated

Sound:

Chart, bar chart

Description automatically generated Chart, bar chart

Description automatically generated Chart, pie chart

Description automatically generated

By contrast, Sound is much less important for my game to meet user needs – it is still important, though less effort can be spent working on it. This means that I will spend minimal time designing sound effects so I will use copyright free ones or generate simple sounds from online tools. This will save time in the project so that I can spend more time on what is more important: the graphics and level design.

The background music is again less important to the users, though it will strongly influence the feel of the game while playing it, so I will ensure to find some copyright free calming music to put for the the background, as that will help the users relax while playing the game.

The sound menu will be very similar to the graphics menu: the users require no more than control over game and music volumes; this will fit easily into a single sound menu screen, which I will implement as part of the menu system

Graphical user interface, application

Description automatically generated

Level Design:

To ensure my levels are fun, engaging and challenging for all users, I need to identify key parameters that must be balanced to make the level accessible to all yet still difficult enough to be interesting.

None of the users want to be stuck on a single level for more that about 20 mins on average and 5 minutes looks like a good balance to ensure the levels remain enjoyable for all, and no one gets frustrated, though some are more patient and will happily play a level for up to half an hour. To meet all needs it would be good to make this variable, though this could take long to implement a system which creates balanced levels of vary sizes.

The users want to have to try a level about 3 times before getting it, so they shouldn’t be too heavy on hazards, though there should still be some to provide the correct level of challenge. All users agree that checkpoints will make the level more playable, so those must be a feature to meet their needs.

Graphical user interface, text, application, email

Description automatically generated

Chart

Description automatically generated

Chart, bubble chart

Description automatically generated

Other Feedback:

Chart, pie chart

Description automatically generated

The title of the game is the first thing a prospective user sees, so it must well represent the game. To accurately represent the game, it will be focused on it being a maze exploration game, as well as being linked to the visual theme of the game. That will entice potential players that are likely to enjoy the game.

Two of the features suggested (player customization and NPC driven story) are both not central to the gameplay, but make the game much more personal, giving each user the feeling of being emotionally connected to their character and their adventure making them more involved in the game.

These features may take a lot of time to implement, especially if they are to be done well, which likely puts them outside of the scope of what I can develop in this time frame.

A scoreboard is also a good idea to implement as that will allow timed competitive runs of the game, though this may be difficult to balance well with the procedural level generation.

A picture containing background pattern

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Interview

### Limitations and Scope

## Proposed Feature List

|  |  |
| --- | --- |
| Feature | Justification |
| Main menu which points to   * Single player * Settings * Leaderboard * EXIT | Allows the user to quickly and easily navigate around all the games functionality |
| Procedurally generated mazes, populated with hazards and objectives automatically | Allows for infinite unique levels to keep the game new and enjoyable. Will take a lot less time to develop than manual levels |
| Ability to change player colour to navigate the maze | Makes the mazes more intricate and challenging to navigate |
| Ability to pick up and place down items to control elements of the maze | Makes the mazes more intricate and challenging to navigate |
| Settings menus for video and audio | Allows the user to configure the game as to make it optimally enjoyable for them |
| Menus must have simple, intuitive buttons and sliders | Enhances ease of use so users can focus on enjoying the game |
| Locally stored Leaderboard | Will allow the user to compete with themselves to beat their high score, making the game more challenging for those who want it |
| 2d top-down camera perspective | Lends itself well to navigating and solving mazes |
| Limited field of view | Hides most of the maze from the user so they must explore it to discover the way out, making the game more challenging and in depth |
| Key game elements highlighted in functional colours | Makes the levels more intuitive as the user is automatically drawn to items and mechanics they need to use |
| Background elements must be relaxing, dark colours | Ensures the overall colour scheme of the game isn’t too bright or startling, which is important to ensure the users can relax by playing the game |
| Ui during gameplay must be minimalistic | Keeps the screen free of clutter which will make it chaotic and stressful to look at. |
| simple animations for interacting with the maze and ui | Adds visual flare that makes the game feel more immersive, allowing the user to relax while playing the game |
| Simple sound effects for interacting with the maze and ui | Provides audible confirmation to the user about what they just did so they know it is important to beating the level |
| Relaxing, playful background music | Creates a calming, immersive atmosphere that ensures the user enjoys the game to full extent without distractions. |

## Why this Solution is Suited to a Computation Solution

This game will have many complex features that must function correctly and interact with each other and the user seamlessly to produce an engaging, fun gaming experience. To do this I will employ computational methods

### Abstraction

The player will walk around the maze, exploring the level, but navigating a real-life maze has a lot of complexities that are unnecessary and will make the game bulky, clunky, and difficult to play. Abstraction allows me to take a way these annoying details while still retaining the original concept intact, but now much easier to interact with and use on a computer screen.

The gameplay will be built upon abstractions, for example, walking around a maze requires putting one foot in front of the other repeatedly to get around a 3d world, but controlling this directly will make the game hard to use and unintuitive, so instead the character controller will simply be the arrow keys which cause the player to move at a fixed rate in that direction on a 2d world. The inventory system will be heavily abstracted, just being a group of items, saving the user the trouble of trying to stuff many things inside a backpack to carry round.

The audio-visual design of the game will be abstracted, the textures being simpler than their real counterparts, with a less crowded colour palette and simpler shapes and less detail. The sound effects will be simpler, comprising of jingles rather than, for example the sound of actual keys being picked up. This serves to prevent viewing and listening to the game from becoming overwhelming to the user, the simplicity making it much more relaxing to use.

Effective use of abstracted design is very important for my game to meet its users’ needs as it allows the game to be intricate and engaging while not becoming overwhelming, laborious, and stressful, which is important while trying to relax and play a videogame.

### Thinking Ahead

To ensure I meet the needs of the stakeholders as effectively as possible, I must carefully plan my game. This requires thinking ahead about how the game will be structured, planning out how it will be designed and how each part will function, reviewing how it should meet the requirements before being implemented.

The game will be planned extensively during the design phase, following a top-down design workflow, where the construction of each feature and how it will interact with all other features will be exactly detailed. This allows me to iteratively review the design to verify it still satisfies the success criteria all the way through development.

Without an effective plan, a project of this scale would quickly become incoherent, with each feature piling on top of the next, making the final solution a complex mess of inter dependent procedures, which would make the game impossible to effectively maintain or iterate on. This highlights how critical thinking ahead is to my game’s success.

### Thinking Procedurally

During playing my game, many events will happen, such as receiving user input, loading assets, processing motion, rendering and animating sprites and displaying that to the screen. The events must be precisely timed to ensure the game behaves as I want it to, or it will become unpredictable.

To handle each sequence of events more easily, the game can be split into smaller, more manageable sub systems; this is Decomposition. There will be many smaller sub systems, such as:

* The game loop
* Asset loading systems
* Sprite rendering
* Maze generation
* Maze population
* Maze rendering
* Menu GUI

Each of these sub systems is a lot smaller and more specific than the game they will coalesce to form, meaning they are much simpler, each implementing only a few algorithms. Each one will be developed in isolation initially with a set of test programs to ensure they meet their functional requirements. This makes debugging much easier as the test programs will repeatably reproduce edge cases, allowing me to understand how my programs behave in tricky situations without struggling to reproduce those situations in the game itself.

### Thinking Logically

During gameplay, the user’s decisions will impact what happens in the game next. This means that I will have to use logical thinking to ensure that certain gameplay paths are only unlocked under the correct conditions.

For example. The player will only be able to go through a door if they collect the correct key: This will require that upon approaching a door the code checks for if the corresponding key is in the player’s inventory, and if it is, the door unlocks, removing its collider box, and if the key is not present, nothing happens

The player controller will require much logical thinking to design. The player must be able to move by taking in control inputs from the keyboard, where the player only move when a key is pressed, and it must decide which direction to move depending on which key it is. The player controller must also consider the environment, ensuring the player only walks on clear ground and never through walls, using conditions to check if there is a wall to the player’s sides before moving, making sure to only move the player if there isn’t a wall in that direction. The walls must also be checked to ensure that they are not the player’s current colour, in which case what don’t need to be collided with.

The main game loop will contain a litany of logic as it must consider what inputs are pressed and the game state to decide what to do with each input, such as checking what game state is currently active, then which parts of that game state have been unlocked, and then which parts of that state are currently being rendered on screen.

### Thinking Concurrently

There are many events that must happen all at the same time in the game; they must be processed concurrently. Concurrency is where the system switches very quickly between multiple processes to give the illusion that they are running in tandem: this will be used ubiquitously throughout my game.

The game loop must handle receiving inputs, updating each sprite, and drawing everything all at the same time as far as the user is concerned, but this can be achieved by checking the inputs, then updating each sprite one by one, then rendering each sprite one by one. This makes the game more playable and engaging than if each even happened one by one like in a text-based adventure game.

The audio system will also utilise concurrent processing as it will play dual channel audio from multiple sources at the same time, all while the game is also running. The background music will be playing from a file on loop in the background while events in game cause different sounds to be played and mixed over top of it.

## Hardware and Software Requirements

|  |  |
| --- | --- |
| Processor: dual core x86 64bit @ 1GHz or better | The game’s code must be executed at a minimum rate to ensure it is fun to play |
| Memory: 2 GB ddr3 | This will allow a minimal operating system build to run as well as the game, so long as it is the only thing running on the system |
| Graphics: 256mb video memory, capable of rendering at 640x480 | The UI will depend on a minimum resolution to render properly and be readable, and this requires a minimum amount of video memory |
| Storage: 500MB available space | All the source code and assets use 500MB of free storage on the system |
| OS: 64 bit Microsoft Windows 10 | Windows is a modern and common operating system providing the required execution environment for the rest of the dependencies |
| Python 3.10 | All my code will be written to be run by the python 3.10 interpreter, so to ensure all syntax is properly processed, python 3.10 is required |
| Pygame 2.1.0 | My code will call pygame 2.1.0 functions, so to ensure that those functions run correctly, pygame 2.1.0 will be a requisite |

## Success Criteria

# B. Design

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

## Systems diagram

# C. Developing the coded solution (“The development story”)

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

# D. Evaluation

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

# Project Appendixes

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

* Complete Code Listing (ESSENTIAL)
* Interview Transcripts
* Meeting notes
* Observation notes or questionnaires