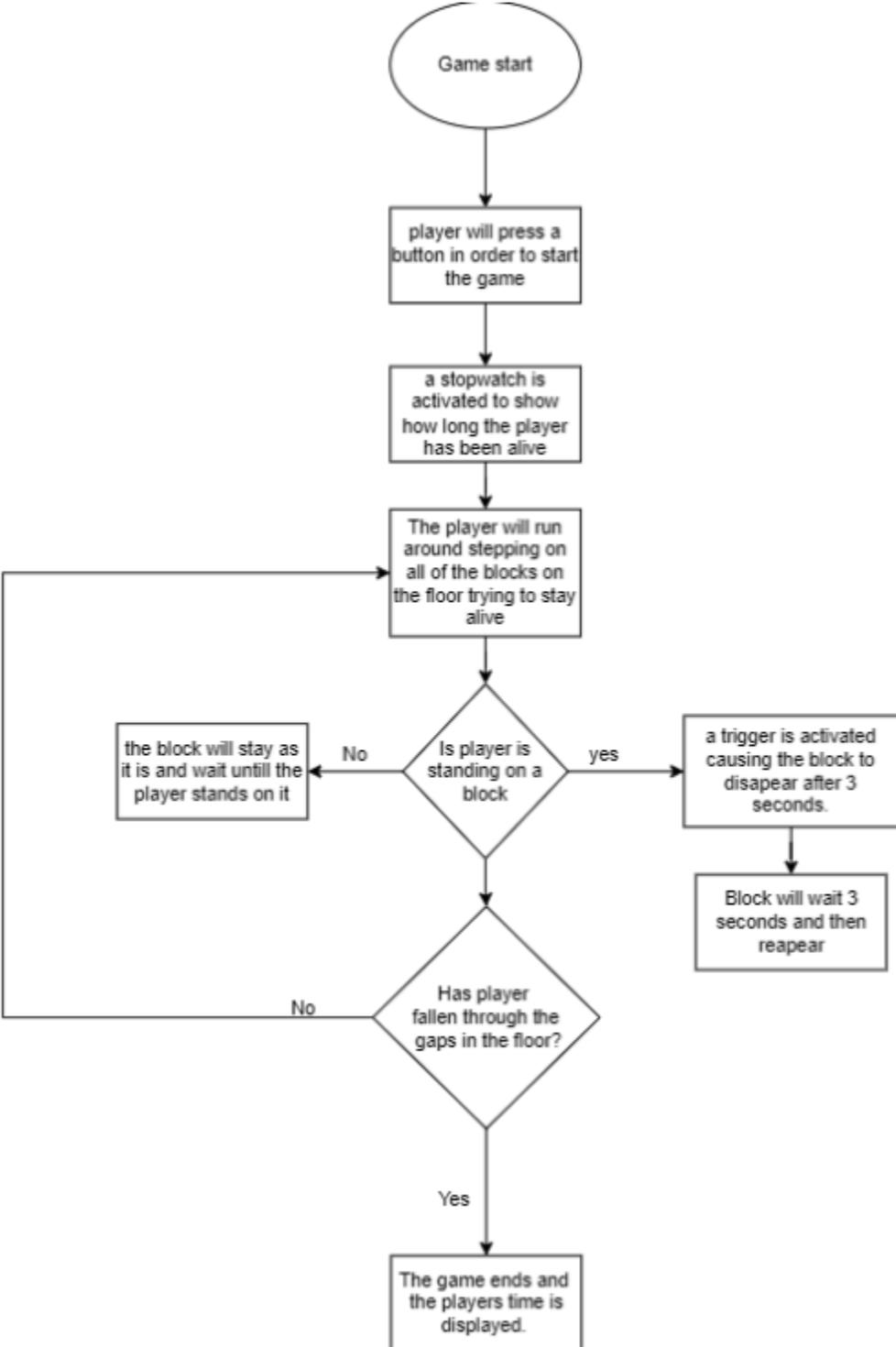


# GAD170.3 Reuben Poole

## Flow Chart



# Pseudo code

In order to make this Game I will need the following scripts:

- Player movement script
- FloorBlock script
- FloorSpawner script
- Timer script
- Score script
- Objective script
- GameOver script
- Ui script

## 1. Start game

- The floor blocks will be spawned in a 50x50 grid. **(FloorSpawner script)**
- One objective will be spawned above one of the floor blocks in the grid. **(FloorSpawner script)**

## 2. Gameplay loop

- The player will run and jump around all of the floor blocks. **(Player movement script)**
- Every floor block that the player steps on will disappear for a certain amount of time and then reappear after a certain amount of time. **(FloorBlock script, Timer script)**

## 3. Objective spawning

- The player must navigate through the floor blocks in order to reach the objective. **(Player movement script)**
- The player will have a certain amount of time to reach the objective otherwise the game will end. **(Objective script, Timer script)**
- Once the player reaches the objective, the score will increase by 1 and a new objective will be spawned above a random floorblock. **(Score script, FloorSpawner script)**

## 4. Game over Conditions

- If the player falls off the edge of the grid the game ends. **(GameOver script)**
- If the player falls down through the space made by the temporarily disabled floor blocks the game will end. **(GameOver script)**
- If the player is not able to reach the objective in time the game ends. **(GameOver script)**

## 5. Restart

- After the game ends, the player will be shown their score and they can choose to restart the game. **(Ui script)**

## Pseudo code update:

My pseudo code predictions on how many scripts my game would use were quite off, however the flow that my game used stayed pretty much the same. In my completed project I used these scripts:

- FloorBlock script
- FloorSpawner script
- Objective script
- Score script
- ObjectiveTimer script
- Timer script
- GameEnder script
- GameOver script
- MainMenu script
- GameSound script
- KeepGoingText script
- PlayerMovement script (GameDevelopment, 2022).
- ThirdPersonCam script (GameDevelopment, 2022).

## Technical Documentation

### FloorBlock Script:

The FloorBlock script is the script for the floor that is instantiated by the FloorSpawner script.

My FloorBlock Script is laid out like this:

---

#### Variables:

At the top of the script I have declared all of the variables that the FloorBlock script will need, these include:

- Difficulty variables.
  - Material references to change the color of each floor block
  - Collider references to make it so the floor blocks can have their collider switched on or off.
  - Timer variables (controls how long the floor blocks disappear for).
  - Show/hide variable.
-

### Start function:

Input none;

Output void;

- sets the material to the normal floor.
- 

### Update function:

Input none;

Output void;

- Calls function that increases difficulty.
- 

### OnTriggerEnter function:

Input Collider;

Output void;

- Swaps material from normalFloor to steppedOnFloor.
  - Starts a coroutine that controls the amount of time it takes for the floor block to disappear.
- 

### ShowFloorBlock function:

Input Bool Show;

Output void;

- Shows/hides the floor block using the object renderer (turning off the object collider so that the player can fall through).
- 

### IEnumerator DisappearAfterTime function:

Input DisappearTime;

Output Coroutine;

- Coroutine waits for disappearTime seconds.
  - Calls ShowFloorBlock(false).
  - Starts another coroutine that makes the block reappear.
- 

### IEnumerator ReappearAfterTime function:

Input reappearTime;

Output Coroutine;

- Coroutine waits for reappearTime seconds.
  - Calls ShowFloorBlock(true).
- 

### IncreaseDifficulty function:

Input none;

Output void;

- Increases the difficulty the longer the game is played.
  - If the time on the timer becomes 10 more than the timePast variable. Reappear time will increase by 1 second and timePast will be set to the current elapsed time.
- 

## FloorSpawner Script:

My FloorSpawner script is laid out like this:

---

### Variables:

At the top of the script I have declared all of the variables that the FloorSpawner script will need, these include:

- References and Prefabs.
  - Variables for the arena grid spawning.
  - A List that all of the floor blocks are added to when they are spawned.
- 

### SpawnFloor function:

Input none;

Output void;

- Instantiates the floor block prefab in a grid to form the arena for the game
  - Uses nested loops to instantiate a line of floor blocks and then spawn another line of floor blocks from each of the instantiated floor blocks.
- 

### PickRandomObjectiveSpawnPlace function:

Input none;

Output void;

- This function picks a random place for the objective to spawn, making sure that it does not spawn above a block that is currently missing.
  - Generates a random number.
  - Goes through the list of floor blocks until a block that isn't hidden is found.
  - Spawns the objective above the floor block that was chosen in the above while loop.
-

DestroyObjective function:

Input none;

Output void;

- Destroys the objective (called after on trigger enter in the objective script).
- 

## Objective Script:

My Objective script is laid out like this:

---

Variables:

At the top of the script I have declared all of the variables that the Objective script will need, these include:

- References.
  - Variables for the trigger cooldown.
- 

OnTriggerEnter Function:

Input Collider;

Output void;

- Displays the Keep Going Text.
  - Plays pickup sound.
  - Resets the objective timer back to whatever number it should be (dependent on what the current score is).
  - Starts coroutine for canTrigger variable.
  - Calls functions that Destroy the objective and then spawn a new one.
  - Increases the score by 1.
  - Calls IncreaseObjectiveDifficulty function.
- 

IEnumerator DontTrigger Function:

Input triggerCooldown;

Output Coroutine;

- Makes it so that the objective can't be triggered by the player for triggerCooldown seconds.
-

## Score Script:

My Score script is laid out like this:

---

Variables:

At the top of the script I have declared all of the variables that the Score script will need, these include:

- References.
  - Score Variable.
- 

Start function:

Input none;

Output void;

- Converts score variable to string so that it can be displayed as text in the UI.
  - Displays the score variable as a string in the UI.
- 

IncreaseScore function:

Input none;

Output void;

- Called in the objective script. (adds 1 to the score each time it is called).
- 

## ObjectiveTimer Script:

My ObjectiveTimer script is laid out like this:

---

Variables:

At the top of the script I have declared all of the variables that the ObjectiveTimer script will need, these include:

- Time variables.
- Variables to do with increasing the difficulty of the objective timer.

- noDecimalPointNumber Variable (Makes the first number in the timer display more cleanly with no decimal point).
  - References.
  - Alpha variable controlling the alpha channel of the time warning Ui Panel.
  - GameEnded Variable.
- 

### Start function:

Input none;

Output void;

- Sets timeout to false and the warning panel to see-through.
- 

### Update function:

Input none;

Output void;

- Sets time left to go down by Time.deltaTime.
  - Displays the time left in seconds and hundredths of a second.
  - Makes the time left display with no decimal points when it is at its highest number of time left (makes the game look nicer in my opinion).  
Starts the time warning when the time left is below 4 - else the time warning panel is see through.
  - Ends the game if the time left is 0 or below.
- 

### TimeWarning function:

Input none;

Output void;

- Called when there is less than 4 seconds left to get to the objective.
  - Causes a panel over the screen to get more red over time to stress the player out and provide feedback to the player.
- 

### IncreaseObjectiveDifficulty function:

Input none;

Output void;

- Makes the objective timer take less time everytime the score goes up by 10.
- 

### ResetTimer function:

Input none;

Output void;

- Resets the objective timer.
  - Called when the player triggers the objective.
- 

### GameOver function:

Input none;

Output void;

- Ends the game.
  - Frees up the mouse so the player can use the menu.
- 

## Timer Script:

My Timer script is laid out like this:

---

### Variables:

At the top of the script I have declared all of the variables that the Timer script will need, these include:

- elapsedTime Variable.
- 

### Update function:

Input none;

Output void;

- Adds Time.deltaTime to the elapsedTime float.
  - Displays elapsedTime in minutes and seconds as a string.
- 

### StopTimer function:

Input none;

Output void;

- Gets the timer script and stops it.
  - This is done when the game ends so that it doesn't keep counting in the displayed results.
-

## GameEnder Script:

My GameEnder script is laid out like this:

---

### Variables:

At the top of the script I have declared all of the variables that the GameEnder script will need, these include:

- References.
  - A boolean variable called canTrigger that makes it so the game ender can only trigger once.
- 

### OnTriggerEnter function:

Input Collider;

Output void;

- Ends the game if the player enters the trigger area.
  - Plays a sound
  - Sets canTrigger to false so that the game over sound doesn't play multiple times.
- 

## GameOver Script:

My GameOver script is laid out like this:

---

### Variables:

At the top of the script I have declared all of the variables that the GameOver script will need, these include:

- References.
  - A boolean variable called gameEnded that makes it so the game can only end once per round.
  - Variables for changing the transparency of the press to continue button (just to make it look good and stand out more)
-

### Start function:

Input none;

Output void;

- Hides the game over UI.
- 

### Update function:

Input none;

Output void;

- Make the pressToContinue button flash a little bit.
  - Loads the menu screen when the right mouse button is pressed.
- 

### ShowStats function:

Input none;

Output void;

- Provides feedback to the player showing their stats on how they did. (score and time they were alive for)
- 

### GameOverSound function:

Input none;

Output void;

- Plays the game over sound.
- 

## **MainMenu Script:**

My MainMenu script is laid out like this:

---

### PlayGame function:

Input Button;

Output void;

- Connected to a button.
  - loads the main game scene.
-

QuitGame function:

Input button;

Output void;

- Connected to a button.
  - Quits the game.
- 

## GameSound Script:

My GameSound script is laid out like this:

---

Variables:

At the top of the script I have declared all of the variables that the GameSound script will need, these include:

- All of the references to the different AudioSources used.
  - gameOverSoundNotPlayed boolean. This is so I can make the game over sound only play once because it was playing once when you fall off the map and once if the timer runs out.
- 

PlaySoundGameOver function:

Input none;

Output void;

- Plays game over sound and then makes it so that the game over sound cant be played again.
- 

PlaySoundJump function:

Input none;

Output void;

- Plays the jump sound.
- 

PlaySoundLand function:

Input none;

Output void;

- Plays the land sound.
- 

PlaySoundPickUp function:

Input none;

Output void;

- Plays the pick up sound.
- 

## **KeepGoingText Script:**

My KeepGoingText script is laid out like this:

---

Variables:

At the top of the script I have declared all of the variables that the KeepGoingText script will need, these include:

- Reference to the keep going TextMeshPro object.
  - Float for how long the keep going text is displayed.
  - Bool for if the keep going text has already been displayed.
- 

Start function:

Input none;

Output void;

- Sets the keep going text to disabled.
- 

ShowKeepGoingText function:

Input none;

Output void;

- If the keep going text has not been shown yet it starts a coroutine that shows the text for a certain amount of time.
-

## IEnumerator ShowKeepGoing function:

Input keepGoingTextShowTime;

Output Coroutine;

- Sets the keep going text to enabled.
  - Waits for keepGoingTextShowTime seconds.
  - Sets the keep going text to enabled.
- 

## Mid-point Reflection

### **Process:**

Due to the fact that I really enjoy the programming aspect of this course I have not had any problems so far with planning my processes for this project. However I have had to use a little bit of restraint in order to ensure that I don't get too carried away and neglect my projects due in other units. In order to manage my priorities for this assignment I have been attempting to implement things into my game that ensure that I tick off all of the requirements of the brief. After I have done all of the brief requirements then I will be able to add more things to my game that I want but are not required. When programming I encounter so many small challenges that I have to overcome. I see all of these challenges simply as an exercise in problem solving and also as a way in which I can develop my knowledge on C# and the unity engine. A particular challenge that I spent quite a bit of time trying to solve was when trying to change the color of my floor blocks when they are stepped on. In the end I was able to overcome this challenge by creating two different materials for the color of the floor block. Although I spent many hours attempting multiple different ways to figure this out. Page 4 of Algorithmic problem solving talks about this style of problem solving. Highlighting that in programming, problem solving is an iterative process where you often need to experiment, refine, and optimize your algorithms to find the best solution (Backhouse, p.4, 2011).

### **Person:**

This project has not required a large amount of communication skills due to the fact that it is not a group project. However due to the challenging nature of programming as a beginner I have had to spend a lot of time working through challenges and solving problems with my lecturer. When responding to feedback I have been very grateful to receive feedback due to the fact that when I ask for help I have often already spent hours attempting to solve the problem on my own and therefore I am so happy to have the problem explained to me by someone with a much more developed understanding of programming and unity than I have.

## Proficiency

In order to complete this project within the desired time frame I have really had to use and develop my skills, techniques, knowledge and use of technology in many areas relating to programming and the Unity engine. For example I used my knowledge of various programming techniques such as creating lists, and using while loops in order to make it so that the objective in my game wont spawn above any of the inactive floor blocks. This was a vital part to add into my game due to the fact that if the objective happened to spawn above an inactive block it would essentially cause the death of the player due to the fact that they would most likely not be able to reach the objective in time. Throughout the process of this project so far I have been able to further my knowledge in programming which is a very important skill that has application in many other professional fields such as, web development, IT, and software development.

## Project-Completion Reflection

### Appraisal:

Overall I am very pleased with how my project turned out, although there are still a few things that I would have liked to add to my game, such as better sound effects and some real 3D models for the objective and the player. I believe that I achieved all of the things required in my code by the brief and furthermore I added quite a lot of stuff to my game that wasn't required. Something that worked really well for me during this project was that I felt I was really starting to develop a better understanding of how to actually do the basics of C# In Unity, I found myself seeing the error message less and less the more I worked on this project which was quite a satisfying feeling. One thing that was quite annoying for me during this project is that I should have probably worked on my technical documentation more during the course of the project instead of completing it all at the end. Due to the fact that this game is far larger than the other games I have created so far I really underestimated the amount of time it would take to complete my technical documentation, and it ended up taking me at least 5 hours to complete. Overall I believe that my game exceeded my expectations, however this is due to the fact that I kept on adding more and more to the game simply because I was enjoying the process.

### Challenges:

One of the main challenges that I faced during this project was that I left my prefabs in my scene the whole time I was creating the game. I was unaware that I wasn't really meant to do this and it led to quite a few issues towards the end of my project, for example, I had my prefabs setup so that they were referencing other scripts in the game. I attempted to resolve this issue and set up the game in a more proper way, however this resulted in so many issues and I could have potentially spent days attempting to get my game back to working order, so therefore I ended up just leaving my prefabs in the scene. If I had more time to get this assignment done I would have definitely fixed these issues however I just don't think that I would have been able to within the time frame of this project. After facing this challenge my teacher

told me about a tool called events, and this prompted me to do some research at home. And found out that events will be a vital tool for me in the future that will help me develop cleaner projects that consist of more separate and easy to understand scripts. From this experience I learned that It is important to do everything the most proper way possible in order to minimize the amount of challenges that I face the more complex my game becomes.

## Future Goals:

In future projects I will improve my skills by ensuring that I continue to research and implement various C# and Unity techniques and tools, especially focusing on developing my knowledge on using events in Unity to make sure that I encounter less major errors and create projects that are of a higher user experience and professional standard. In my next project something that I will attempt to do differently is that I will try to comment out my code as I go along instead of doing it all at the end, I believe that If I had of been doing this throughout the whole project I would have been able to greatly improve my workflow due to the fact that it would have been much easier and quicker to understand where I was in my code without having to read the actual code. In future projects of this nature something that I will definitely be repeating is using GitHub in order to allow me to work on my projects both at home and on campus. GitHub was so helpful and it allowed me to be able to show my teacher any issues I was having with my code without having to download my work onto google drive.

## References

Backhouse, R. (2011). Algorithmic problem solving. Chichester, England: Wiley.

GameDevelopment, D. (2022). THIRD PERSON MOVEMENT in 11 MINUTES - Unity Tutorial [YouTube Video]. In *YouTube*.

<https://www.youtube.com/watch?v=UCwwn2q4Vys>