­­­Computer Science Programming Project – PoolGame

*Internal Deadline: end of Autumn Term 2024 (Friday 20th December)*

Contents

[Notes 1](#_Toc179444788)

[Key 1](#_Toc179444789)

[Writeup 1](#_Toc179444790)

[Program 1](#_Toc179444791)

[Analysis of the problem 2](#_Toc179444792)

[Problem identification 2](#_Toc179444793)

[Stakeholders 2](#_Toc179444794)

[Research the problem 2](#_Toc179444795)

[Specify the proposed solution 6](#_Toc179444796)

[Design of the solution 9](#_Toc179444797)

[Decomposing the problem 10](#_Toc179444798)

[Describe the solution 10](#_Toc179444799)

[Describe the approach to testing 10](#_Toc179444800)

[Developing the solution 10](#_Toc179444801)

[Iterative development process 10](#_Toc179444802)

[Testing to inform development 10](#_Toc179444803)

[Evaluation 10](#_Toc179444804)

[Testing to inform evaluation 10](#_Toc179444805)

[Success of the solution 10](#_Toc179444806)

[Describe the final product 10](#_Toc179444807)

[Maintenance and development 10](#_Toc179444808)

[Bibliography 10](#_Toc179444809)

# Notes

## Key

Notes (to remove in the final version)

First draft (to expand upon)

## Writeup

* USE TABLES FOR THINGS
* Use Big O for measurements
* Identify success criteria that can be definitively tested and used to determine the success, or otherwise, in developing that feature of the solution
* Abstraction (easier to code, easier for CPU to run)
* Bibliography at end
* Add table of contents after it’s done
* Assign each success criteria a name, e.g.:

1. [placeholder]
2. [placeholder]

## Program

* USE ASSERTS
* Tag-in system to keep track of everyone’s scores even if they aren’t playing (e.g. at the start of each game, it asks the names of the players (allowing them to choose from a list of pre-existing names (maybe created at start-up) or make a new name))
* Tutorial
* Supports default controllers (Xbox, PlayStation) with a setting (maybe with a popup on the start screen) to add controllers by recording inputs and assigning them to certain controls
* Persistent data storage (maybe using .txt or .csv)
  + Settings
  + Scores
  + Different ‘user profiles’ have different saved settings
* Popup when the game is first installed (or when a new user enters) to enable various a11y features

# Analysis of the problem

## Problem identification

There currently are few group activities that are available to most people; most are either too expensive, slow to pick up on, or too repetitive. A solution to this problem could bring people together more (where other methods may be out of their reach) or just provide an easier and simpler way for groups to be entertained.

A computational approach is perfect for this problem as it can easily be customized to accommodate whoever’s using it, whereas real-world entertainment methods often can’t meet everyone’s needs (e.g. someone with mobility issues might have a hard time with sports, someone with very little money might not be able to afford a lot of entertainment methods that already exist).

My solution could be taken almost anywhere if it can run on a laptop, and there would be less risk of losing/damaging parts of it in transit. It’s also cheaper for end-users since they just need to install it rather than buying parts or paying fees, making it accessible to a wide range of people regardless of their financial situation (with some exceptions). In addition, my solution can be more scalable than a real-world game because no more equipment or space is needed for extra players (especially if only one keyboard and mouse/controller is needed).

## Stakeholders

### Casual gamers (***STK.1***)

These will likely be a large portion of my solution’s demographic, so they should be accommodated for as well; they range in age from children to adults. They play games to socialise and have fun, rather than to compete or challenge themselves, so my solution needs to be intuitive, relaxed, and have local multiplayer. Among them are some pre-teen children, so my solution will need to be suitably kid-friendly as well.

### Casual competitive gamers (***STK.2***)

These are still a large portion of the demographic so suitable mechanics should be introduced so that they can be entertained as well. They will need some sort of score-keeping system and a leader board to be implemented.

### Individuals with accessibility needs (***STK.3***)

A part of the problem my solution is trying to solve is the lack of availability of games to some groups of people, so my solution should be able to accommodate most their needs. They may need a narrator, colour-blind modes, etc.

### Notes

Due to the range of conflicting requirements, toggles may need to be available to alter how some mechanics work for different audiences.

I will be using ***STK.1***, ***STK.2***, and ***STK.3*** to refer to my stakeholders. If something is applicable to a particular stakeholder, I will write (***STK.x***) after it. (Later on, these will be replaced with ***SC.x*** for success criteria)

### Testers

My solution will be tested by its stakeholders where possible to reveal how suitable it is for them and for any other feedback they may have. Where this is not possible, I will be sure to research other solutions to problems with similar demographics to accommodate their needs. This will help me improve my solution by adding, tweaking, or removing features.

Some testers may have computer science experience and as such will be able to provide direct feedback on the source code. They may be able to evaluate my solution with a more technical lens, increasing the chance of finding bugs or exploits.

## Research the problem

[Add Images of each existing solution]

### Existing solutions

I will be looking at existing solutions to parts of my problem (or similar problems) and noting features to apply to my solution and ones to be mindful of (i.e. features that could be added but might not be suitable). If a similar feature is present multiple existing solutions, I’ll only write it down in the first game it appears to remain concise.

#### Gang Beasts

|  |  |
| --- | --- |
| Feature which could be applied to my solution | Justification |
| Simple menus and a “ready up” system | These ensure that players can start games quickly and when everyone is ready. The simple menus also make it easier for players to change settings and exit games, reducing the need for online guides and streamlining player’s experiences. (***STK.3***) |
| Simplistic environments and a minimalistic graphical user interface | These make it clearer for players to identify what they should be focusing on during a game, without being distracted by anything meaningless. (***STK.3***) |
| Environmental hazards | These create a more interesting and chaotic environment for players to engage with, reducing the boredom that comes from replaying the same thing over and over again. (***STK.1***)  They also provide different chances for players to combat each other in simple, unique ways without requiring new controls that may be hard to input and/or learn. (***STK.2***) |
| No inappropriate imagery | This makes the game suitable for all ages, allowing younger children to play. (***STK.1***) |
| Each player is assigned a colour and name and are allowed to pick an outfit for their character | This helps players avoid confusing who’s who mid-game even if they have some vision difficulties. (***STK.3***) |
| Using a keyboard, players can slow down or speed up the game with + or – respectively. | This make it more manageable for players with slower response times to be on an equal footing with faster players. (***STK.3***) |
| Both controller and keyboard/mouse support, with intuitive controls (e.g. space bar/A as jump and WASD/joysticks for movement) and an option to remap them in the settings. | This makes it easier for players to learn to play the game without needing time to remember specific controls. (***STK.1***)  While it may give less freedom to extreme competitive gamers (not casual competitive gamers), they aren’t a part of my stakeholders so I shouldn’t compromise the overall playability of my game for them. |
| Up to 4 players able to play at any one, with heavy interactivity between players and a score system. | This encourages players to be more competitive and obstruct each other, while keeping the playing field fair. (***STK.2***) |

|  |  |
| --- | --- |
| Features to be mindful of in my solution | Explanation |
| It has simple and complex combo moves | They are not explained to player very clearly, so most people either don’t know they exist or stumble into them accidentally. However, the game isn’t built off of these, so the simple moves are enough to play. |
| It revolves around fighting, so it could be seen as unsuitable for some audiences. | It skirts around this by using cartoon violence (i.e. no blood or gore) and it is presented in a light-hearted way. But players can still be ‘knocked out’ in more gory ways like by being pushed into a meat grinder or eaten by a shark. |

#### Jackbox Party Pack Series

|  |  |
| --- | --- |
| Feature which could be applied to my solution | Justification |
| No reliance on reaction time, only having a timer to make sure players submit their answers relatively quickly | This bypasses the need for a game-speed modifier option, |
| Each ‘Party Pack’ has different games to choose from when playing, keeping players entertained even if they’ve played it before. |  |
| Simple controls (via a mobile device) to submit answers with an on-screen mobile keyboard or the touch screen (e.g. for drawing). |  |
| The tone of the game is set by the players, meaning much younger groups likely won’t see any inappropriate content whereas older groups have the freedom to have as much fun as they want. There is also an option in the settings menu to suppress potentially inappropriate content. |  |
| Up to 8 players can play most of the games within each pack, with a score system (or sometimes an elimination system). |  |

|  |  |
| --- | --- |
| Features to be mindful of in my solution | Explanation |
| A connection to the internet is required for players to submit their answers | With no in-person alternative due to the nature of the game, it makes playing it in-person require every player to have a mobile device. If players are in a place with no internet, it can’t be played (especially if some players can’t use mobile data). |

#### Golf With Your Friends

|  |  |
| --- | --- |
| Feature which could be applied to my solution | Justification |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Features to be mindful of in my solution | Explanation |
|  |  |
|  |  |

Features which could be applied to my solution:

* Builds off an existing and well-known game (crazy golf), so most new players are already familiar with the rules.
* Players can interact with each other (by knocking into other player’s golf balls to send them off course).
* A player’s golf ball is always in view of them and is customisable, with colours, a name, and a trail. This helps them avoid confusing who’s who mid-game even if players have some vision difficulties.
* The game is scored in terms of how many shots it takes for a player to sink their golf ball, so players with slower reaction times have an equal chance of winning.
* The settings menu has options to modify gameplay (e.g. toggle ball collisions, alter gravity), so players can customize their experience to how they’d prefer.

Features to be mindful of in my solution:

* Some courses can be quite challenging to complete (e.g. precise balancing or aiming), so easier courses should be available (or some form of handicap).

#### Mario Kart 8 Deluxe

|  |  |
| --- | --- |
| Feature which could be applied to my solution | Justification |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Features to be mindful of in my solution | Explanation |
|  |  |
|  |  |

Features which could be applied to my solution:

* Assistance modes (e.g. auto-accelerate and smart steering) may help players who are less skilled or able keep up with other players.
* Item boxes add an element of randomness to the game and enable players who are behind to still have a chance to catch up (especially with items like the blue shell, which temporarily stuns the player in first place).
* Difficulty options (e.g. item balancing and bot difficulty levels) allow for a more customisable experience and more fun for players who aren’t looking for too hard of a challenge.

Features to be mindful of in my solution:

* It requires a Nintendo Switch to play, which not everyone has access to (especially because it, like most other consoles, isn’t as practical for work as something like a laptop, computer, or phone)
* By its nature as a racing game, it relies on reaction time which may be challenging for those with certain physical disabilities or coordination difficulties.
* The items may seem unfair, particularly if a player is hit multiple times with detrimental ones, so some balances may be necessary to make sure less people get frustrated.

#### Minecraft: Bedrock Edition

|  |  |
| --- | --- |
| Feature which could be applied to my solution | Justification |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Features to be mindful of in my solution | Explanation |
|  |  |
|  |  |

Features which could be applied to my solution:

* Cross-platform multiplayer allows for more people to play together even if they don’t all have the same devices.
* Accessibility features (e.g. text-to-speech) help more people play on a similar level, rather than be disadvantaged.
* Variable graphics settings allow simultaneously for players with lower-end machines to play and for those with higher-end machines to fully utilise theirs for improved graphics.
* The menus are large and well-spaced with an easy-to-read default font, making them easy to navigate.

Features to be mindful of in my solution:

* It has a steep learning curve and is open-ended which could be overwhelming for newer players, since they are thrust into the world with no objective and many mechanics explained naturally in-game (e.g. potion brewing or finding the stronghold) aren’t. There are many guides online, but players may not realise they even exist.
* While it is optimised for many devices, some lower end ones still have performance issues.

#### Pool (video game and analogue versions)

|  |  |
| --- | --- |
| Feature which could be applied to my solution | Justification |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Features to be mindful of in my solution | Explanation |
|  |  |
|  |  |

Features which could be applied to my solution:

* The objective and rules are simple and well-known, so newer players are more likely to be able to play quickly.
* Turn-based play allows players to take their time when planning their moves and removes any reliance on reaction time.
* Its reliance on real-world physics makes how the ball moves more intuitive to players.
* While players typically share a cue ball, they can still disrupt each other’s strategies by leaving the cue ball in hard-to-shoot-from locations (such as next to the 8-ball or surrounded by balls of the opponent’s suit) which adds to the strategy and interactivity between players.

Features to be mindful of in my solution:

* Its gameplay loop can feel repetitive, with little variation. Whereas games like one-pocket or nine-ball pool are meaningfully different, they can still feel repetitive if played enough.
* It relies on player’s precision for some difficult shots, which may make it harder for experienced players to hold their own.

#### Accessibility Guidelines

While this isn’t an existing solution outright, it’s a guideline to one. I’ll keep it in mind when designing my game, so that it is built with accessibility in mind rather than having it patched on later (which could be difficult and/or time-consuming). For this I will be predominantly using [gameaccessibilityguidelines.com](https://gameaccessibilityguidelines.com/) because it lists a lot of features that I might not think of while designing. It separated features into categories (e.g. motor, hearing) and by how easy they are to implement.

On top of using that website, I’ll try to avoid needing to patch accessibility problems by not causing them in the first place. For example, instead of implementing a colour-blind mode, my game won’t rely on only colour for players to differentiate different objects from each other.

### Features of my proposed solution

My solution will be a simulated game of pool so that most new players can play without having to learn a new game. I have chosen pool because its premise is simple (i.e. hit more balls into pockets than your opponent to win) so it is easy to understand for new players. Due to its simplicity, it can be expanded upon to suit certain group’s preferences (e.g. adding obstacles, adding an ‘items’ equivalent, having more than one cue ball, etc.). Also, it will be a turn-based game, meaning that people with slow reaction times or other similar difficulties won’t be at a meaningful disadvantage while playing.

It will be able to be played by 2+ players at a time so that larger groups have less people watching on the side-lines being bored, and smaller groups can play too. A single player mode isn’t relevant to the stakeholders, so it isn’t likely to be added.

It will have simple menus to help players start games and find any relevant settings quickly. Where possible, there should be multiple ways for players to receive information and differentiate objects in the game (e.g. colours and patterns to differentiate between different object balls). There should also be controller support (on top of keyboard and mouse) because it helps people with needs re-map the controls to use with their specialised hardware.

#### Limitations

For larger groups, if it’s a turn-based game and people spend a while waiting for their turn. Or if not it’s a real-time game and, since it’s local multiplayer, it would have too many inputs into 1 device to function. It may work if online multiplayer was an option, but the main problem my game is trying to solve involves players being together in-person.

[placeholder]

## Specify the proposed solution

With the above idea of how to approach the problem in mind, I have outlined the requirements and features of my game. I have categorised the features to help keep in the scope of the project.

### Requirements

#### Functional

|  |  |  |
| --- | --- | --- |
| Requirement | Justification | Success criteria |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |

#### Non-functional

|  |  |  |
| --- | --- | --- |
| Requirement | Justification | Success criteria |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |

#### Hardware

|  |  |  |
| --- | --- | --- |
| Requirement | Justification | Success criteria |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |

#### Software

|  |  |  |
| --- | --- | --- |
| Requirement | Justification | Success criteria |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |
| Fill this in | Fill this in | Fill this in |

### Features

#### Essential (i.e. stakeholder requirements)

|  |  |
| --- | --- |
| Feature | Justification |
| Reasonably accurate simulated physics | If the game isn’t a realistic enough adaptation of analogue pool, it may feel uncomfortable or unintuitive to play (potentially putting off some players). If it’s overly realistic it will be slower to run on low-end machines, so it will be less accessible to people whose computers don’t meet the requirements.  This would exclude a section of my game’s stakeholders, so I will need to thread the line between overly realistic and not realistic enough. |
| Local multiplayer (2+ players, turn-based) | This allows gatherings to be fully entertained while playing my game, without people waiting on the side lines too long.  It should be noted that with a large number of players (4+), players may have to wait their turn anyway because of how long a round could get (but that’s better than them not being able to play at all). |
| A Score system | If the game is scored in-game, it allows STK.2 to compete without needing pen-and-paper scorekeeping. |
| Accessibility features | These will help those with specific needs have an easier and more fun time playing my game. Since people with accessibility needs are a portion of my stakeholders, these are high on my list. |

#### Desirable

|  |  |
| --- | --- |
| Feature | Justification |
| Controller support (on top of keyboard and mouse) | Controllers are preferred by a lot of STK.1 because they can be used from further away (e.g. on a sofa while the game is on a TV) and are easy to move around (when players are cycling who’s currently playing).  Also, some people have specialised controllers to help with accessibility needs. Adding controller support provides an easier way for these people to re-map the controls to their controller so they can play. |
| Control re-mapping | This enables the above feature (controller support) to be used more effectively by people with specialised hardware. It also allows anyone else who feels uncomfortable with the controls the ability to customise them to suit their needs. |
| Minimalistic/simplistic graphics (including menus) | Simple menus would help players change settings easier and start game sessions quicker, streamlining their experience. Simple graphics would make it easier for players to keep track of what’s important mid-game and for players with lower-end computers to play without performance issues. |
| Multiple/clear ways for players to receive information and differentiate objects in the game | This would reduce potential confusion for players, lessening the chance of mistakes made by players like potting opponent’s object balls. This issue would be exemplified in players with vision/reading difficulties, so having ‘back up’ methods for viewing objects. |
| For example:   * Colours and patterns to differentiate between different object balls * High contrast between borders |

#### Optional

|  |  |
| --- | --- |
| Feature | Justification |
| Bot players | While this feature isn’t a requirement from stakeholders, it could still make it nicer for smaller/one-person groups so that they can play with more ‘players’. |
| Difficulty options (if applicable) | This allows for a more customisable experience and more fun for players who aren’t looking for too hard of a challenge. But it can only be implemented if there are already features that can be tweaked (e.g. item boxes or bots). |
| Options for different game modes / toggleable features | This is a way of adding some variety to my game. If players either don’t find pool fun or have played my game too much they could still hopefully find some enjoyment in it. |
| For example:   * Enabling environmental hazards (allowing for more ways for players to make use of / be challenged by their environment without requiring any new controls to learn) * Enabling item boxes (or equivalent) so that players who are behind have a chance to catch up.   Multiple cue balls, enabling interactivity between players (by knocking into other player’s cue balls to send them off course). |

#### Out-of-scope/Excluded

|  |  |
| --- | --- |
| Feature | Justification |
| Online multiplayer | The main focus of the game is for local multiplayer, so it wouldn’t meaningfully benefit the majority of the stakeholders. Also, online multiplayer can be difficult and time-consuming to implement. It would take up a lot of development time for a feature that most people wouldn’t use. |
| Multiple platforms | This amount of development would take a lot more time to implement than I have available. My game will be limited to an application that can be installed on PCs and other similar hardware. |
| Inappropriate imagery, or wording | My game must be suitable for as wide of an age range as possible, so anything that isn’t suitable for young children shouldn’t be added. |

“As part of the analysis candidates are expected to identify suitable measurable success criteria. In many cases the success criteria identified are far too generic often referring to ease of use or aesthetic considerations. The success criteria will be used together with evidence of testing to establish the effectiveness of the solution. It is important the student identifies success criteria that can be definitively tested and used to determine the success, or otherwise, in developing that feature of the solution. The table above is a part of the success criteria for a game. The student identifies a measurable feature, establishes how that feature can be tested and provides justification for that feature as part of the success criteria.”

Limitations of mobile apps: hard for many people to use/see at one time

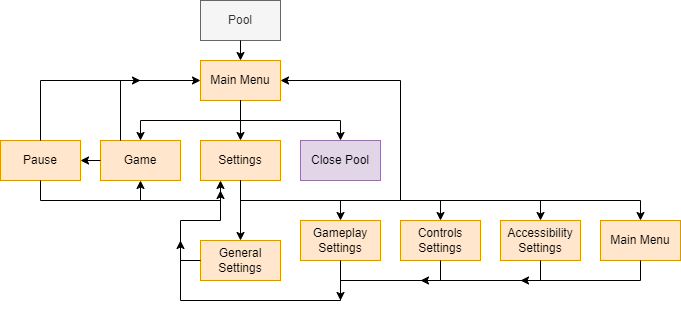
# Design of the solution

Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.

## Decomposing the problem

I’ve broken down my game into different components that can be brought together to form the game. Each component is associated with a screen in the game and handles the underlying logic of that screen. They should be structured so that they are all connected (directly or indirectly) so that each of them is quick and easy to get to for a streamlined experience.

Splitting up different screens also ensures that processes (e.g. collision detection) aren’t run pointlessly while in another screen, which will improve overall performance.

Overall, the connections between screens should look something like this:

Orange rectangles represent screens, arrows represent the screens that can be accessed from a screen. The diagrams for individual screens will only have the screens they lead to, not the ones that lead to them, because otherwise it would look too cluttered.

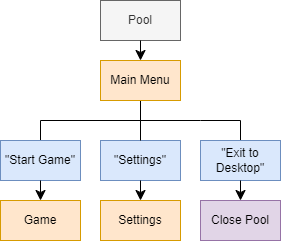
### Menus

Most of the screens will be menus, mainly featuring buttons, dropdown menus, and sliders. Buttons will be used to move between screens (e.g. “Settings” buttons), dropdown menus will be used for changing discrete settings (e.g. controls), and sliders will be used to change continuous settings (e.g. volume).

#### Main Menu

The Main Menu screen is the screen that will initially be shown when my game is opened, it is linked to by any button labelled “Exit to Main Menu”. Its main function is to be a hub for most of the other screens, while allowing users to exit the program. There will be 3 buttons, each button’s destination screen will be given in brackets (if applicable):

* “Start Game” (Game Screen) with default [or previously-saved] gameplay settings
* “Settings” (Settings)
* “Exit to Desktop” – closes the program, requires a confirmation to prevent accidental exits.



Blue rectangles represent buttons, with their display text in quotes.

###### Justification

The Main Menu should be displayed when the game starts so that players can access settings before they start a game. This should allow players to modify settings to enhance their experience as soon as they boot up the game if they need.

Because of the Main Menu’s utility, it is linked to by every screen (except Settings’ sub-screens) so that it can be accessed easily.

It should be noted that because closing the program and exiting a game in my program both can come under the term “Exit Game” (despite being different), “Exit to Desktop” will be used to close the program and “Exit to Main Menu” will be used to exit a game and go to Main Menu.

#### Settings

The Settings screen is linked to by any button labelled “Settings” and is much like the Main Menu screen in that both are hubs to reach other screens. This screen will have 5 buttons:

* “General” (General Settings)
* “Gameplay” (Gameplay Settings) [– if the Settings screen is opened during a game, this menu’s settings (or at least some of them) will not be changeable.]
* “Controls” (Controls Settings)
* “Accessibility” (Accessibility Settings)
* “Back” (Main Menu) – in Settings’ sub-menus, back will be used to go back to Settings

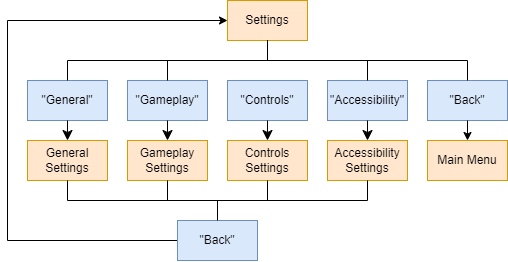
###### Sub-menus

The settings screen has four different sub-menus that show more specific settings (as listed above)

###### Justification

This screen gives users the ability to customise their experience. It’s separated into different menus so that users can quickly find the settings they are looking for, instead of having to search through a big list.

[Gameplay Settings’ settings will not be fully changeable mid-game because it could lead to some game-breaking effects.]



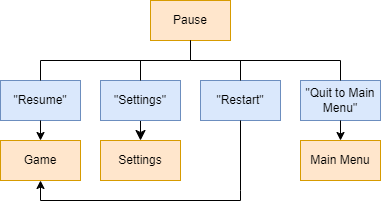
#### Pause

The Pause screen will popup when a certain key is pressed during a game. It will have the following buttons:

* “Resume” (Game) – Resumes the current game
* “Settings” (Settings)
* “Restart” (Game) – Initialises a new game, requires a confirmation to prevent accidental restarts
* “Exit to Main Menu” (Main Menu) – requires a confirmation to prevent accidental exits

###### Justification

The pause screen will allow players to pause the game if they need to take a break or do something else. It will also let players exit or restart games smoothly and change settings mid-game to test them out or correct them without restarting a game.



### Game

The Game screen holds most of the substance of my game, it displays pool and handles the logic behind it (e.g. collisions). It is split into three phases: aim, move, and game end; this is to avoid it checking for collisions or similar things while it isn’t needed to save memory.

Confirmation if ever exiting/restarting game

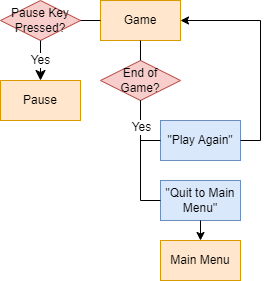
3 phases

* Aim – the current player aims and shoots the cue ball
* Move – the cue ball moves
* Game end

Resets stats at Game End or the “Restart” button rather than the start of the game, so that going to the pause menu and back doesn’t automatically restart

###### Justification

[placeholder]



TODO:

* UPDATE STRUCTURE DIAGRAM
* ~~A structure diagram illustrating the problem decomposition.~~
* Explanation of all the game objects.
* Algorithms in pseudocode for the methods in the scripts.
* Usability features.

1. Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.

## Describe the solution

|  |  |  |
| --- | --- | --- |
| Computational Method | Where it can be used | Justification |
| [placeholder] | [placeholder] | [placeholder] |
| [placeholder] | [placeholder] | [placeholder] |

Class Diagram Template:

|  |
| --- |
| ClassName |
| - attributeNamePrivate: AttributeType |
| - methodNamePrivate(parameter1, parameter2)  + methodNamePublic(parameter1, parameter2) |

|  |  |  |
| --- | --- | --- |
| Class | Explanation | Justification |
| ClassName | [placeholder] | [placeholder] |
| ClassName | [placeholder] | [placeholder] |

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Method | Explanation | Justification |
| ClassName | MethodName | [placeholder] | [placeholder] |
| MethodName |
| ClassName | MethodName | [placeholder] | [placeholder] |
| MethodName |

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Attribute | Explanation | Justification |
| ClassName | AttributeName | [placeholder] | [placeholder] |
| AttributeName |
| ClassName | AttributeName | [placeholder] | [placeholder] |
| AttributeName |

.NET is intermediate code

* A structure diagram illustrating the problem decomposition.
* Explanation of all the game objects.
* Algorithms in pseudocode for the methods in the scripts.
* Usability features.
* Data structures to be used.
* Input validation.

1. Explain and justify the structure of the solution.
2. Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.
3. Describe usability features to be included in the solution.
4. Identify key variables / data structures / classes justifying choices and any necessary validation.

## Describe the approach to testing

* Input validation.
* Test data to be used during the development of the coded solution.
* Test data to be used post-development.

1. Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.

# Developing the solution

## Iterative development process

## Testing to inform development

# Evaluation

## Testing to inform evaluation

## Success of the solution

## Describe the final product

## Maintenance and development

# Bibliography

* <https://www.imdb.com/title/tt10950132/parentalguide>
* <https://support.jackboxgames.com/hc/en-us/articles/15794756085015-How-many-players-can-join-each-game>