Computing A level Coursework by Arman Shaikh

Table of Contents

[Analysis 10](#_Toc98495346)

[Introduction 10](#_Toc98495347)

[Stakeholders 10](#_Toc98495348)

[Zach Tracy 10](#_Toc98495349)

[Harry Anthony 10](#_Toc98495350)

[Alyvia Mur 10](#_Toc98495351)

[Computational approach 11](#_Toc98495352)

[Research 12](#_Toc98495353)

[Dino Runner 12](#_Toc98495354)

[Fortnite 12](#_Toc98495355)

[Strike force heroes 13](#_Toc98495356)

[Summary 14](#_Toc98495357)

[Hardware and software minimum requirements 15](#_Toc98495358)

[Hardware 15](#_Toc98495359)

[Software 15](#_Toc98495360)

[Peripheral requirements 15](#_Toc98495361)

[Inputs 15](#_Toc98495362)

[Outputs 15](#_Toc98495363)

[Player requirements 16](#_Toc98495364)

[Limitations 17](#_Toc98495365)

[Programme limitations 17](#_Toc98495366)

[Player limitations 17](#_Toc98495367)

[Game balancing limitations 17](#_Toc98495368)

[Crash prevention limitation 17](#_Toc98495369)

[Success criteria 18](#_Toc98495370)

[Menu 18](#_Toc98495371)

[Movement 18](#_Toc98495372)

[Inputs 18](#_Toc98495373)

[Gameplay 18](#_Toc98495374)

[HUD 19](#_Toc98495375)

[Design 20](#_Toc98495376)

[What the client expects 20](#_Toc98495377)

[Inputs 20](#_Toc98495378)

[Outputs 20](#_Toc98495379)

[Design 20](#_Toc98495380)

[Processes 20](#_Toc98495381)

[Usability features 20](#_Toc98495382)

[Decomposing 23](#_Toc98495383)

[Algorithms 24](#_Toc98495384)

[Tank-repair-kits appearing and used algorithms. 24](#_Toc98495385)

[Player input management 25](#_Toc98495386)

[Enemy sprite movement 26](#_Toc98495387)

[Algorithm for ammo kit appearance 28](#_Toc98495388)

[Algorithm to get the top scores of the player. 29](#_Toc98495389)

[Databases clean up. 31](#_Toc98495390)

[Login 32](#_Toc98495391)

[Password verification 33](#_Toc98495392)

[Account Deletion 34](#_Toc98495393)

[Keeping user data 35](#_Toc98495394)

[Tokens 38](#_Toc98495395)

[Data storage structures 39](#_Toc98495396)

[Testing method 40](#_Toc98495397)

[Stage 1 40](#_Toc98495398)

[Account making 40](#_Toc98495399)

[Login 40](#_Toc98495400)

[Phase 1 40](#_Toc98495401)

[Phase 2 40](#_Toc98495402)

[Phase 3 40](#_Toc98495403)

[Phase 4 41](#_Toc98495404)

[Phase 5 41](#_Toc98495405)

[Stage 2 41](#_Toc98495406)

[Menu 41](#_Toc98495407)

[Settings editor 41](#_Toc98495408)

[Shop 41](#_Toc98495409)

[Stage 3 42](#_Toc98495410)

[Game loop 42](#_Toc98495411)

[Phase 1 – movement 42](#_Toc98495412)

[Phase 2 – shells 42](#_Toc98495413)

[Phase 3 – hook 42](#_Toc98495414)

[Phase 4 – enemy 43](#_Toc98495415)

[Phase 5– kits 43](#_Toc98495416)

[Stage 4 43](#_Toc98495417)

[Further tests that are done post-development 43](#_Toc98495418)

[White box 43](#_Toc98495419)

[Black Box 43](#_Toc98495420)

[Software development 44](#_Toc98495421)

[Prototype 1 44](#_Toc98495422)

[Day 09/05/2021 44](#_Toc98495423)

[Main v1 44](#_Toc98495424)

[Login v1 45](#_Toc98495425)

[Registration v1 46](#_Toc98495426)

[globals v1 49](#_Toc98495427)

[White box testing 51](#_Toc98495428)

[Summary 52](#_Toc98495429)

[Day 10/05/2021 53](#_Toc98495430)

[Login v2 53](#_Toc98495431)

[globals v2 56](#_Toc98495432)

[White box testing 57](#_Toc98495433)

[Summary 57](#_Toc98495434)

[Day 11/05/2021 58](#_Toc98495435)

[Main v2 58](#_Toc98495436)

[Login v3 59](#_Toc98495437)

[globals v3 61](#_Toc98495438)

[Menu v1 63](#_Toc98495439)

[Shop v1 64](#_Toc98495440)

[Scores v1 67](#_Toc98495441)

[White box testing 69](#_Toc98495442)

[Summary 70](#_Toc98495443)

[Day 12/05/2021 71](#_Toc98495444)

[Main v3 71](#_Toc98495445)

[globals v4 72](#_Toc98495446)

[Menu v2 74](#_Toc98495447)

[Scores v2 75](#_Toc98495448)

[Settings v1 76](#_Toc98495449)

[Delete v1 78](#_Toc98495450)

[Game v1 81](#_Toc98495451)

[HUD v1 82](#_Toc98495452)

[White box testing 83](#_Toc98495453)

[Summary 85](#_Toc98495454)

[Day 13/05/2021 86](#_Toc98495455)

[Game v2 86](#_Toc98495456)

[globals v5 86](#_Toc98495457)

[HUD v2 90](#_Toc98495458)

[Player v1 92](#_Toc98495459)

[Registration v2 94](#_Toc98495460)

[Settings v2 97](#_Toc98495461)

[Shell v1 99](#_Toc98495462)

[White box testing 100](#_Toc98495463)

[Summary 100](#_Toc98495464)

[Day 14/05/2021 100](#_Toc98495465)

[Game v3 100](#_Toc98495466)

[globals v6 102](#_Toc98495467)

[Shell v2 104](#_Toc98495468)

[Zombie v1 106](#_Toc98495469)

[White box testing 107](#_Toc98495470)

[Summary 108](#_Toc98495471)

[Day 15/05/2021 109](#_Toc98495472)

[Main v4 109](#_Toc98495473)

[Login v4 111](#_Toc98495474)

[Registration v3 113](#_Toc98495475)

[globals v7 116](#_Toc98495476)

[Menu v3 119](#_Toc98495477)

[Controls v1 120](#_Toc98495478)

[Scores v3 121](#_Toc98495479)

[Settings v3 123](#_Toc98495480)

[Delete v2 125](#_Toc98495481)

[Game v4 128](#_Toc98495482)

[HUD v3 130](#_Toc98495483)

[Shop v2 133](#_Toc98495484)

[Player v2 136](#_Toc98495485)

[Shell v3 138](#_Toc98495486)

[Hook v1 140](#_Toc98495487)

[Zombie v2 141](#_Toc98495488)

[MedKit v1 143](#_Toc98495489)

[AmmoKit v1 145](#_Toc98495490)

[White box testing 146](#_Toc98495491)

[Summary 148](#_Toc98495492)

[Summary of prototype 1 149](#_Toc98495493)

[References for prototype 1 152](#_Toc98495494)

[Prototype 2 153](#_Toc98495495)

[Day 24/05/2021 153](#_Toc98495496)

[Main v5 153](#_Toc98495497)

[Credits v1 157](#_Toc98495498)

[Login v5 158](#_Toc98495499)

[Registration v4 159](#_Toc98495500)

[Menu v4 160](#_Toc98495501)

[Scores v4 161](#_Toc98495502)

[Settings v4 162](#_Toc98495503)

[Delete v3 165](#_Toc98495504)

[Game v5 166](#_Toc98495505)

[HUD v4 166](#_Toc98495506)

[Shop v3 169](#_Toc98495507)

[Player v3 170](#_Toc98495508)

[Zombie v3 170](#_Toc98495509)

[White box testing 173](#_Toc98495510)

[Summary 174](#_Toc98495511)

[Day 25/06/2021 176](#_Toc98495512)

[Main v6 176](#_Toc98495513)

[Credits v2 180](#_Toc98495514)

[AmmoKit v2 180](#_Toc98495515)

[MedKit v2 180](#_Toc98495516)

[Hook v2 180](#_Toc98495517)

[HUD v5 181](#_Toc98495518)

[Player v4 184](#_Toc98495519)

[Zombie v4 187](#_Toc98495520)

[White box testing 191](#_Toc98495521)

[Summary 191](#_Toc98495522)

[Day 26/06/2021 192](#_Toc98495523)

[Main v7 192](#_Toc98495524)

[AmmoKit v3 196](#_Toc98495525)

[MedKit v3 197](#_Toc98495526)

[Zombie v5 199](#_Toc98495527)

[White box testing 203](#_Toc98495528)

[Summary 203](#_Toc98495529)

[Summary for prototype 2 204](#_Toc98495530)

[References for prototype 2 207](#_Toc98495531)

[Prototype 3 208](#_Toc98495532)

[Day 14/07/2021 208](#_Toc98495533)

[Main v8 208](#_Toc98495534)

[Login v6 212](#_Toc98495535)

[Registration v5 215](#_Toc98495536)

[Lore v1 218](#_Toc98495537)

[Controls v2 219](#_Toc98495538)

[Credits v3 220](#_Toc98495539)

[Delete v4 221](#_Toc98495540)

[HUD v6 224](#_Toc98495541)

[Menu v5 229](#_Toc98495542)

[Player v5 231](#_Toc98495543)

[Scores v5 235](#_Toc98495544)

[Settings v5 237](#_Toc98495545)

[Shell v4 239](#_Toc98495546)

[ShellFire v1 241](#_Toc98495547)

[ShellLarge v1 243](#_Toc98495548)

[Shop v4 244](#_Toc98495549)

[globals v7 247](#_Toc98495550)

[White box testing 250](#_Toc98495551)

[Summary 251](#_Toc98495552)

[Day 24/07/2021 252](#_Toc98495553)

[Main v9 252](#_Toc98495554)

[globals v8 255](#_Toc98495555)

[Login v7 260](#_Toc98495556)

[Registration v6 262](#_Toc98495557)

[Menu v6 268](#_Toc98495558)

[Settings v6 269](#_Toc98495559)

[Delete v5 273](#_Toc98495560)

[Controls v3 276](#_Toc98495561)

[Lore v2 279](#_Toc98495562)

[Shop v5 281](#_Toc98495563)

[Scores v6 284](#_Toc98495564)

[Game v6 287](#_Toc98495565)

[HUD v7 289](#_Toc98495566)

[Player v6 293](#_Toc98495567)

[Shell v5 297](#_Toc98495568)

[ShellFire v2 299](#_Toc98495569)

[ShellLarge v2 300](#_Toc98495570)

[Zombie v6 300](#_Toc98495571)

[Hook v3 302](#_Toc98495572)

[AmmoKit v4 303](#_Toc98495573)

[MedKit v4 304](#_Toc98495574)

[BossZombie v1 304](#_Toc98495575)

[White box testing 306](#_Toc98495576)

[Summary 310](#_Toc98495577)

[Summary of prototype 3 312](#_Toc98495578)

[References for prototype 3 315](#_Toc98495579)

[Evaluation 316](#_Toc98495580)

[Post development testing 316](#_Toc98495581)

[Registration and login 316](#_Toc98495582)

[Menu 317](#_Toc98495583)

[Shop 318](#_Toc98495584)

[Lore 318](#_Toc98495585)

[HUD 319](#_Toc98495586)

[Settings 319](#_Toc98495587)

[Usability 320](#_Toc98495588)

[Post development Black box testing 322](#_Toc98495589)

[Bug 1 322](#_Toc98495590)

[Bug 2 322](#_Toc98495591)

[Bug 3 323](#_Toc98495592)

[Bug 4 323](#_Toc98495593)

[Bug 5 324](#_Toc98495594)

[Success criteria check 326](#_Toc98495595)

[Stakeholder evaluation 330](#_Toc98495596)

[Zach Tracy 330](#_Toc98495597)

[Harry Anthony 330](#_Toc98495598)

[Alyvia Mur 330](#_Toc98495599)

[Further development 331](#_Toc98495600)

[Limitations 331](#_Toc98495601)

[Transcript 331](#_Toc98495602)

[Main.tscn 331](#_Toc98495603)

[Login.tscn 336](#_Toc98495604)

[Registration.tscn 339](#_Toc98495605)

[Menu.tscn 346](#_Toc98495606)

[globals.gd 349](#_Toc98495607)

[Shop.tscn 353](#_Toc98495608)

[Scores.tscn 357](#_Toc98495609)

[Controls.tscn 359](#_Toc98495610)

[Credits.tscn 362](#_Toc98495611)

[Lore.tscn 362](#_Toc98495612)

[Settings.tscn 365](#_Toc98495613)

[Delete.tscn 369](#_Toc98495614)

[Game.tscn 373](#_Toc98495615)

[HUD.tscn 376](#_Toc98495616)

[Player.tscn 381](#_Toc98495617)

[Zombie.tscn 386](#_Toc98495618)

[BossZombie.tscn 389](#_Toc98495619)

[MedKit.tscn 392](#_Toc98495620)

[AmmoKit.tscn 393](#_Toc98495621)

[Hook.tscn 394](#_Toc98495622)

[Shell.tscn 396](#_Toc98495623)

[ShellFire.tscn 398](#_Toc98495624)

[ShellLarge.tscn 400](#_Toc98495625)

[Video Evidence 402](#_Toc98495626)

[Login 402](#_Toc98495627)

[Registration 402](#_Toc98495628)

[Menu 403](#_Toc98495629)

[Shop 403](#_Toc98495630)

[Scores 404](#_Toc98495631)

[Controls 404](#_Toc98495632)

[Credits 404](#_Toc98495633)

[Lore 404](#_Toc98495634)

[Settings 404](#_Toc98495635)

[Delete 405](#_Toc98495636)

[Game 405](#_Toc98495637)

[HUD 405](#_Toc98495638)

[Player 406](#_Toc98495639)

[Zombie 406](#_Toc98495640)

[BossZombie 406](#_Toc98495641)

[Main 406](#_Toc98495642)

[globals 407](#_Toc98495643)

[Bibliography 407](#_Toc98495644)

# Analysis

## Introduction

My plan is to make a 2D, top-down, shooter game with a personal leaderboard. The player will log in locally; after login, the game, will see and update the personal leaderboard; the game then shows the menu so the player can change settings like the settings on the key bindings or to clear their personal leaderboard. The user also could start playing the age; acquiring a higher score the longer they play until they are defeated; every enemy killed will give a certain amount of in-game cash, the game then sees if they need to change the player’s high score. Once done it goes back to the menu until the user picks another option. If it is the shop, the person can use the currency to upgrade the player indefinitely, like increasing health, ammo capacity, the health you get back from tank-repair-kits or ammo kits, or increasing the turrets fire rate or bullet speed. They can then close the shop and once done they can close the game.

## Stakeholders

### Zach Tracy

Zach likes shooter games like ‘Fortnite’ and ‘PUBG: Battlegrounds’ however to play them you would need a decent internet connection; however, he lives in the countryside so has poor internet speeds. He wants something to quench his boredom but also wants a feel of accomplishment in beating a highscore. So as a stakeholder he wants a continuous shooter game that he can play till it bores him and allows him to see his scores to compare it to others. He will probably use it when the internet is down or slow so it will stop his boredom.

### Harry Anthony

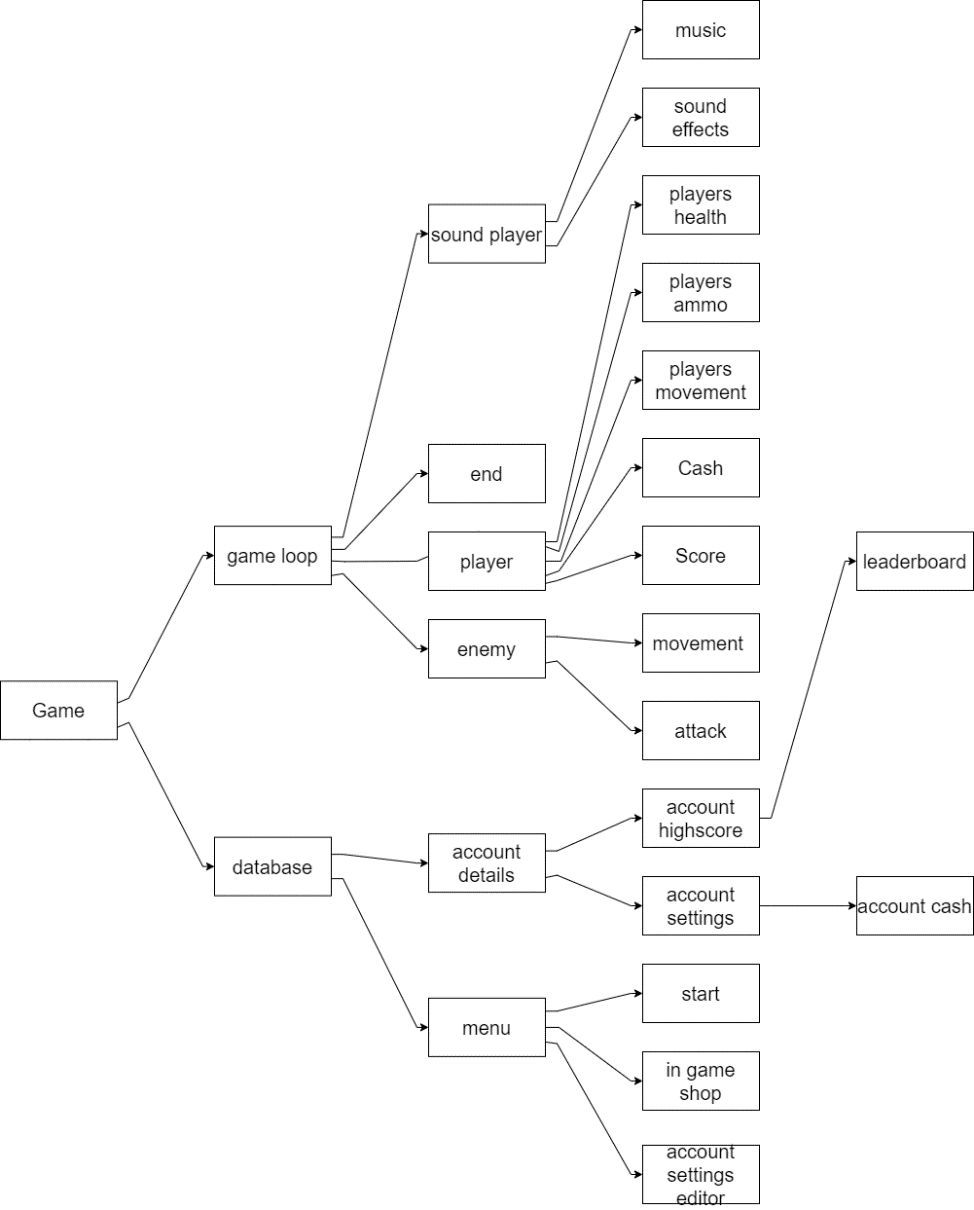
Harry likes the thrill, the fast pace and ramping of difficulty of a game like ‘dine runner’ by Google so he wants a game that allows him to feel the rush and get into the zone with to beat his previous high score however he like new games. As a stakeholder, he wants a game that ramps up its difficulty as it goes on until it is nearly impossible to play. He will use the game to improve his reaction speed as the game gets harder and to fulfil his adrenaline needs as his mind zones in to beat his previous high score.

### Alyvia Mur

Alyvia likes progression games where you can improve your player, but she also likes 2d shooter games but has never found one she likes. As a stakeholder she wants some way of progressing the player. She will probably use it to challenge her peers by progressively improving her player and then comparing their scores.

## Computational approach

The current professional approach to making a game is to decompose it down into sizable chunks making it easier to code like using Object Oriented Programming (OOP) to make everything modular E.G. a class for the sprites, or home screen. All the modules though, would still be interconnected. This can be seen in the graph, with all the hard parts abstracted away so then the developer can code easier.

The code for the database and the game loop is separate as they do not interact at all through gameplay and vice versa.

The player module has the players health as it is restarted every new game by checking the cached max health drawn from the database on boot-up or on edit. The player’s health, ammo and movement are all under it as they are all dependent or affected by the player. I abstracted how the player affects the game.

The enemy and its two children are separate from the player as they are AI-controlled; they can only do 2 things: movement and attack. Movement is a recurring task, while attack only happens once. I abstracted the sprite changes as it moves.

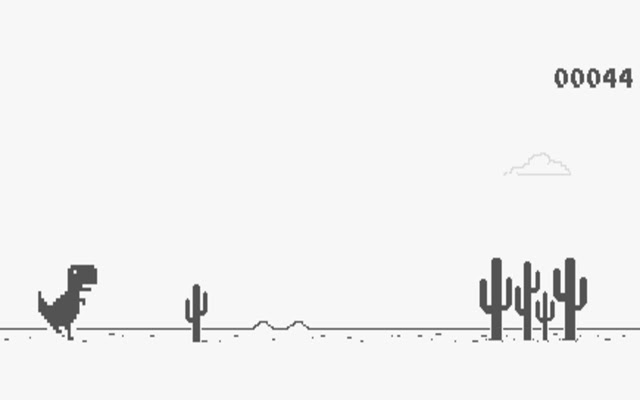
The sound player is in the game loop as it is used only in the game to make noises; it is affected by the cached setting; the setting is gathered from the database on boot-up or when the player signs in.

The database is only used to store details like settings or upgrades on every account and stores their passwords in a secure hash table.

The stuff that will not be programmed in is the global leaderboard as it requires a host server.

## Research

### Dino Runner

For research, I looked up how similar games that did this, and I came across the dinosaur jump game by Chrome. It is an offline game that is shown on the chrome browser if the person tried to connect to a website without an internet connection, the game is simple; you play as a dinosaur that must jump are duck under obstacles(like flappy birds but easier) the game gets faster as it goes on making it harder and harder to play until the person fails where it shows your current score and the high score from your current instance in playing. It has most of the thing I was looking for:

* Continuous play – the end-user wants this.
* Offline – the end-user wants this.
* 2D – easier to code.
* Randomness – makes the game more enjoyable as there is an element of chance in it.
* Had a score/highscore system – The end-user wants it.

But it also:

* Did not have a leaderboard – The end-user expected this.
* Was not a shooter – The end-user wanted this.

What I hope to incorporate is the continuous play feature using the increasing speed of enemies to end the game like dino jump, first to satisfy the end-users demand for it to kill time and to make sure it does not get boring. I will also incorporate the game’s mechanics of random obstacles with the enemies starting randomly on the edge of the map; this will open the game to randomness so that every gameplay is the same. The game also has a high score system which shows your highest score in that instance of playing, you get the score by the longer you play but in my game it will on the number of enemies killed, this will be incorporated to make the player feel a sense of they won if they get a higher score than before.

### Fortnite

Thus, I looked for a game that had the 2 things I was looking for and found Fortnite; it is an online shooter game that people play on their lonesome or with friends; it has gained huge popularity during the last few years. The player will play against 100 other people and try to get to be the last one standing, to make sure the game does not go on indefinitely they have a world wall that lowers HP rapidly when beyond it, the wall continuously shrinks drawing the last players into the zone until one player is left.

It has features like:

* Leaderboards – this can increase enjoyment as you can see if you did better than last time or better than a friend.
* Is a shooter – my game will be a shooter.
* Having a health bar – makes the game easier for the player as they can see at a glance how much health they have left, so they can go, and get a tank-repair-kit.
* Having an ammo bar – makes the game easier for the player as they can see at a glance how much ammo they have left, so they can go, and get an ammo kit.
* Change weapon – gives variety, and choice of what weapon to pick, as they say, that choice is the spice of life.
* Change keybindings – make the game more user-friendly as some people like to play the game differently.
* Lore – creates a story.
* Fullscreen capable – some people like playing on full screen.

But it does not have:

* Continuous play – the end-user wants it.
* Offline capabilities – the end-user wants it offline.
* 2D mode – I am not proficient enough to make it 3D.

For Fortnite, I wish to incorporate its HP bar so that the player can easily find out what their status is by the length of the line. It is a good idea to be able to change key bindings for stuff in settings like Fortnite, making the game more platform friendly to people who want to change keybindings. Lore is what makes a game more interesting than the rest in its genre, to do this I will have a short lore piece that a person can read from the menu; this might be about how the player is stuck in a tank with its engines broken but with the turret still working, they must survive as long as they can; they start with a full barrel of ammo and can get more from ammo drops from rescue planes that cannot get to close or face the wrath of the zombies. The player also gets dropped tank-repair-kits/ammo kits at random from them, to get the kits they must launch the hook to drag it to them.

### Strike force heroes

To soothe these deficiencies, I found Strike Force Heroes; it is an online multiplayer game where the character plays as a person who must kill the opposition; you can do it in campaign mode or free for all. You can also upgrade your weapons for higher-level fights. The features that were useful from it are:

* 2D – my game will be a 2d one.
* Shooter – my game will be a shooter.
* Has a health bar – makes the game easier for the player as they can see at a glance how much health they have left, so can go, and get a tank-repair-kit.
* Has an ammo bar – makes the game easier for the player as they can see at a glance how much ammo they have left, so they can go, and get an ammo kit.
* Player upgrade – this can give a hook to the player to just play longer to upgrade their character.

But it also has deficiencies in being:

* Online – game must be offline.
* Not having a leaderboard – this can increase enjoyment as you can see if you did better than last time or better than a friend.

I wish to incorporate the way you can upgrade your players and have an in-game shop using in-game cash to upgrade the player; this will make the player feel that they are actually doing something instead of endlessly grinding for nothing. It is 2d and uses the mouse to control the gun. I might have a setting where you can control the player’s movement and trigger with the mouse. The game also has a bar for ammo (seen in the bottom right), I will incorporate this to make gameplay easier. I could also make it optional to have either a bar or straight number for the HP and ammo for convenience for the player.

### Summary

Summary of essentials to be added to the game to make it functional and fun.

|  |  |
| --- | --- |
| What to add | Reason |
| Full screen capabilities. | Most games have it |
| Make is so all keybinds can be changed. | Most games have it, and it makes the game more user-friendly. |
| Allow customization of tank for a player. | Makes the game more personal |
| Have text to speech when controls are shown. | Makes it more user-friendly for non-English readers. |
| A continues shooter game. | Main idea for game and a stakeholder requirement. |
| Add lore of the player. | Gives the game some depth. |
| Add an HP and ammo bar. | Most games have it. |
| Themed music to play in the background. | Makes it more enjoyable. |
| A score system. | To make it more competitive between users. |
| A progression system, e.g. upgrading the player with a in game currency. | Makes it enjoyable as the user feels a surge of accomplishment of beating an old score with their better player and as it is a stakeholder’s requirement. |
| The ramping of difficulty as the game continues. | A stakeholder requirement and its needed as continuous game, it will become dull if it does not become harder. |
| Themed music. | Increases tension in the game if the music is themed e.g. having separate main music and game music. |
| Have a secure password. | Passwords are important. |
| Add a boss. | Bosses makes the game challenging and a way to show what zombie wave you’re on. |
| Add confirmation boxes. | Most games have this to decide big setting changes. |
| Different ammo types. | Makes it more strategy based as it allows the player to think of when to use different ammo types |

### Hardware and software minimum requirements

#### Hardware

|  |  |  |
| --- | --- | --- |
| Type | Item | Reason |
| CPU (Central Processing Unit) (Intel) | Intel Core 2 Duo E8400 3.00GHz or newer | Minimum requirement for the Godot engine. |
| GPU (Graphics Processing Unit) (Nvidia) | NVIDIA GeForce 6200 or newer | Minimum requirement for the Godot engine. |
| GPU (Graphics Processing Unit) (AMD) | AMD Radeon HD 7000 series GPU (Graphics Processing Unit) or newer | Minimum requirement for the Godot engine. |
| GPU (Graphics Processing Unit) (Intel) | Intel 3rd generation (Ivy Bridge) series GPU or newer | Minimum requirement for the Godot engine. |
| RAM (Random Access Memory) | 4GB or more | Minimum requirement for the Godot engine. |
| Storage | 200MB or more | Minimum requirement for game file size. |

#### Software

|  |  |  |
| --- | --- | --- |
| Type | Item | Reason |
| OS (Operating System) (Windows) | Windows 7 or newer | Minimum requirement for the Godot engine. |
| OS (Operating System) (Apple) | MacOS 10.10 or newer | Minimum requirement for the Godot engine. |
| OS (Operating System) (Linux) | Linux (64-bit or 32-bit x86) | Minimum requirement for the Godot engine. |
| OpenGL(minimum) | OpenGL 2.1 | Minimum requirement for the Godot engine. |
| OpenGL(recommended) | OpenGL 3. 3 | Minimum requirement for the Godot engine. |
| Pixel shader(minimum) | 3.0 | Minimum requirement for the Godot engine. |
| Vertex shader(minimum) | 3.0 | Minimum requirement for the Godot engine. |
| Pixel shader(minimum) | 3.0 | Minimum requirement for the Godot engine. |
| Vertex shader(minimum) | 3.0 | Minimum requirement for the Godot engine. |

### Peripheral requirements

#### Inputs

* A functional keyboard to control the game character during the game and to sign in.
* A functional mouse to access and use the menu.

#### Outputs

* Speaker/earbuds if they want to hear sounds.
* A good monitor to display gameplay.

### Player requirements

* To play, the person must have 2 functional hands to interact with the computer.
* Must have 2 functional eyes to see the game.
* Must have a functional brain – process all the sensory inputs and outputs.
* Must have a computer to play on.
* Must have time to play.
* Optional – functional ears to hear audio.

## Limitations

### Programme limitations

1. Cannot have a global leaderboard as it requires a server that costs money to run.
2. No multiplayer setting, it is offline cannot be done also this is for killing time.
3. The game must be offline, because of the target audience.
4. Must meet requirements to play.
5. The game must have a type of method to show the HP and ammo.
6. Must have a keyboard to play.
7. Must have a mouse to play.

### Player limitations

1. Cannot code for blind people.
2. Cannot be played by people with no functional fingers.
3. Cannot be played by dead people.
4. Cannot be played on a tablet.
5. Cannot use touch-screen interactions.

### Game balancing limitations

1. Cannot move the player only rotate it; the game character is stationary; it is part of the plot of the engine breaking down and only the turret working, and the player stuck in the wasteland with zombies for miles.
2. Must make the different ammos have different timers as ones that stay for a long time are game breaking.

### Crash prevention limitation

1. Have a limit of enemies on the screen at once, as it will lag the game down to have too many and make it too hard if the movement sensitivity is not high enough.
2. Must delete all scores that are not in the top 5 to not clog up space; it will take too much storage to have too much.
3. Must not allow more than 10 accounts per instance of the game, so people do not make an infinite amount, filling the storage until there is a computer crash.
4. Cannot rapidly fire, it will make the game too buggy if that happens.

## Success criteria

### Menu

1. People can make up to 10 accounts on each game instance, so people do not make an infinite amount, filling the storage until there is a computer crash.
2. The player can exit the game from the Heads-Up Display(HUD).
3. The player can pause the gameplay.
4. Make sure login system works.
5. Make sure the password system is secure
6. The person can change input settings (key bindings).
7. The person can change volume settings.
8. The person can delete all records of their scores.
9. The person can change the sensitivity of movement.
10. The person can reset all progress.
11. The person can make it full screen.
12. The person can change if they want the HP to be either a bar or as a number or both.
13. The person can change if they want the ammo to be either a bar or as a number or both.
14. The person can change and delete their account.
15. The player can see the amount of in-game cash they have in one gameplay.
16. The player can see the total in-game cash they have on the menu.
17. The player will be able to see their top 5 scores.
18. The player will be able to see the top 5 scores from all accounts.
19. The player can upgrade themselves in the in-game shop with in-game cash.
20. Themed music must play in certain scenes.
21. Some lore that is interesting

### Movement

1. The enemy moves towards the player and faces the player.
2. The player can rotate
3. The shell travels in the direction fired
4. The hook travels in the direction fired

### Inputs

1. Upon gameplay, the rotation speed is affected by the sensitivity set in the settings.
   1. player.rotation = player.rotation + (input(right)\*sensitivity)
   2. player.rotation = player.rotation – (input(left)\*sensitivity)
2. The player can fire a tank shell upon the assigned key binding.
3. The player can fire a hook upon an assigned key binding.

### Gameplay

1. The player’s health must decrease if the enemy touches them and then the enemy disappears.
2. The enemy must disappear if touching shell/explosion and increase score and in-game cash.
   1. score = (distance\_of\_mob\_from\_player\*5) + score
   2. Formula for cash: cash = cash + (cash\_upgrade\*5)
3. The enemies speed and damage done to the player must increase linearly during one gameplay and reset after it.
   1. velocity = base\_velocity\*ticks
   2. HP = HP – (base damage\*ticks)
4. The enemy must appear randomly around the player.
5. Tank-repair-kits and ammo kits must appear randomly on the screen.
6. Tank-repair-kits and ammo kits must disappear when touching the hook and increase the appropriate bar.
   1. If (tank-repair-kit\_collison) true{HP = HP +(base\_HP\_increase\*tank-repair-kit\_multiplier)} false{Count =+ 1}
7. Tank-repair-kits and ammo kits must disappear in a set amount of time if they are not used.
   1. If (Count > random.num(1,10)
8. The player is defeated when health is 0.
9. The player can see enemies.
10. The player can see the playable character.
11. The player can interact with the game.
12. The kits will appear randomly.
13. Player can change ammo type
14. Must have at least 2 variants of enemies: normal and bosses.

### HUD

1. The player can see their ammo and HP levels.
2. The player can see the score they got in one gameplay.
3. Must be able to pause and quit the game.
4. The player must be able see to the amount of cash they got.

# Design

## What the client expects

### Inputs

* Must have a menu to change settings and start the game.
* Must have a way to log in.
* The player must be able to control the character by the designated keybinds (the arrow keys or ASDW key).
* The player can pause the game by pressing the specific key bind.

### Outputs

* The game is visible on a graphical screen.
* The game shows players high scores.
* The game display must be smooth.

### Design

* The game must increase in speed the longer it is played.
* The music should be aesthetically pleasing.
* It must be intuitive to play.
* Must be fast paced.
* Characters should not be basic shapes.

### Processes

* The motions of the player and the enemy should be fluid.
* The ammo and tank-repair kits should appear randomly.
* The enemy’s speed should increase linearly.
* Must be able to go through all the high scores and find the top 5 of all and the top 5 of the people playing it.
* Must be able to update the scores of the players.

### Usability features

|  |  |  |
| --- | --- | --- |
| What | Why | Example |
| Buttons | They allow the player to interact with the game |  |
| Menu | They allow the player to start any of the different scenes |  |
| GUI | To allow the game to be seen |  |
| Text boxes | To allow the player to login or register by putting in their login details |  |
| Sound | Compliments the game | <https://www.youtube.com/watch?v=Fxf_iC7hY8I>  Sad music to play when the player is dead. |
| Tank | Player’s avatar |  |
| Shells | Projectiles used |  |
| Explosion | Explosion of shell when it hits the target | A picture containing metalware, hook, clipart  Description automatically generatedA picture containing plant, flower, lamp  Description automatically generatedA picture containing plant, flower, lamp, clipart  Description automatically generatedIcon  Description automatically generated |

## Decomposing

The game has been decomposed into 2 sections, the database, and the game loop. The database module is further divided into menu and account details; this is done as it is easier to edit the database from the menu and as the account details do not have any code. The game loop is subdivided into 4 sections: sounds, player, enemy and end it is done this way as the enemy and player only interact once they touch each other. The sound engine just plays music in the background and sound effects, these could be the turret firing, the shell exploding, the zombie walking, the player losing, the turret tuning, replenishing health by tank-repair-kits, replenishing ammo by ammo kits, losing HP, tank-repair-kit appearing, ammo kit appearing and changing settings.

## Algorithms

### Tank-repair-kits appearing and used algorithms.

The tank-repair-kits appearing are random; first, it takes the size of the game window; then it gets a number smaller that fits the coordinates for it to appear. Then it uses a function from the sound engine to activate the sound effect of the tank-repair kit arriving, once it waits 30 seconds, if it gets touched by the hook it then the hook launches a function to increase the health and another to cause the tank-repair-kit to hide and reset. If in 30 seconds it is not touched, then it disappears for a randomly set amount of time to appear again.

**While Online == True:**

**Sprite. Tank-repair-kit.hide()**

**Count = 0**

**While GameLoop == True:**

**MaxX = OS.window\_size.x**

**MaxY = OS.window\_size.y**

**medX = random.randomint(1,MaxX)**

**medY = random.randomint(1,MaxY)**

**Sprite.Tank-repair-kit.show(medX, medY)**

**collision = collionCheckhook(tank-repair-kit)**

**if collision == 1:**

**Sprite.Tank-repair-kit.hide(MaxX,MaxY)**

**HP =+ 20\*HP\_multiplier**

**Count =+ 0.1**

**delay(0.1)**

**if Count == 15:**

**Sprite.Tank-repair-kit.hide(MaxX,MaxY)**

**delay(random.randint(1,20))**

### Player input management

The player can only rotate and launch projectiles. In the game the player rotates and if the specific keybinds is pressed then launches the procedure to launch either the hook or the shell while also starting the procedure for the sound effect of the specific item. The speed of the rotation is based on the sensitivity that can be changed in the menu and is saved in the database.

**player\_rotation = 0**

**rightRotationInput = database.RightRotationKeybind() #** like ‘right’ or ‘right\_arrow’

**leftRotationInput = database.LeftRotationKeybind() #** like ‘left or ‘left\_arrow’

**hookInput = database.HookKeybind()**

**turretInput = database.TurretKeybind()**

**sensitivity = database.sensitivity()**

**sprite.turret.hide()**

**cash = 0**

**while GameLoop= = True:**

**MaxX = OS.window\_size.x**

**MaxY = OS.window\_size.y**

**turretX = MaxX/2**

**turretY = MaxY/2**

**sprite.turret.show(turretX, turretY)**

**if OS.input(rightRotationInput):**

**player\_rotation += 1\*sensitivity**

**if OS.input(leftRotationInput):**

**player\_rotation -= 1\*sensitivity**

**if OS.input(hookInput):**

**sprite.hook.hookLaunch()**

**sound.hookLaunch()**

**if OS.input(turretInput):**

**sprite.shell.ShellFire()**

**sound.ShellFire()**

### Enemy sprite movement

There will be one main enemy scene that when instanced would randomly appear around the screen and start moving towards the player and randomly augments its base colour. When touching the shell, it causes it to explode, hitting more targets; this also causes the enemy to hide, and it increases the score times the time multiplier. The movement of the enemy increases linearly from the start of the game. Each enemy will have its own code, so they are independent of each other.

**Function Enemy#2():**

**Sprite.Enemy#2.show()**

**MaxX = OS.window\_size.x**

**MaxY = OS.window\_size.y**

**turretX = MaxX/2**

**turretY = MaxY/2**

**enemy#2X = sprite.enemy#2.GetX**

**enemy#2Y = sprite.enemy#2.GetY**

**if enemy#2Y>0 and enemy#2X>0:**

**enemy#2\_direction = 180+Tan-1(enemy#2X/ enemy#2Y)**

**sprite.enemy#2.X= enemy#2X-(1\*ticks)**

**sprite.enemy#2.Y= enemy#2Y-(1\*ticks)**

**if enemy#2Y<0 and enemy#2X>0:**

**enemy#2\_direction = 270+Tan-1((enemy#2Y\*-1)/ enemy#2X)**

**sprite.enemy#2.X= enemy#2X-(1\*ticks)**

**sprite.enemy#2.Y= enemy#2Y+(1\*ticks)**

**if enemy#2Y<0 and enemy#2X<0:**

**enemy#2\_direction = Tan-1((enemy#2Y\*-1)/ (enemy#2X\*-1))**

**sprite.enemy#2.X= enemy#2X+(1\*ticks)**

**sprite.enemy#2.Y= enemy#2Y+(1\*ticks)**

**if enemy#2Y>0 and enemy#2X<0:**

**enemy#2\_direction = 270+Tan-1((enemy#2X\*-1)/ enemy#2Y)**

**sprite.enemy#2.X= enemy#2X+(1\*ticks)**

**sprite.enemy#2.Y= enemy#2Y-(1\*ticks)**

**collision = collionCheckShell(Ememy#2)**

**if collision == 1:**

**sprite.Enemy#2.hide(MaxX,MaxY)**

**score = score + (1\*ticks)**

**cash =+ 5**

**return**

**if collision == 2:**

**sprite.Ememy#2.hide(MaxX,MaxY)**

**HP =- 1**

**if enemy#2Y==0 enemy#2X>0:**

**enemy#2\_direction = 270**

**if enemy#2Y==0 enemy#2X<0:**

**enemy#2\_direction = 90**

**if enemy#2Y>0 and enemy#2X=0:**

**enemy#2\_direction = 180**

**if enemy#2Y<0 and enemy#2X=0:**

**enemy#2\_direction = 0**

**delay(0.5)**

**ticks=+0. 5**

### Algorithm for ammo kit appearance

The appearance and usage of the ammo kit is the same as the med kit.

**While Online = True:**

**Sprite. AmmoKit.hide()**

**Count = 0**

**While GameLoop == True:**

**MaxX = OS.window\_size.x**

**MaxY = OS.window\_size.y**

**ammoX = random.randomint(1,MaxX)**

**ammoY = random.randomint(1,MaxY)**

**Sprite.AmmoKit.show(medX, medY)**

**collision = collionCheckhook(ammokit)**

**if collision == 1:**

**Sprite.AmmoKit.hide(MaxX,MaxY)**

**ammo =+ 20\*ammo\_multiplier**

**Count =+ 0.1**

**delay(0.1)**

**if Count == 15:**

**Sprite.AmmoKit.hide(MaxX,MaxY)**

**delay(random.randint(1,20))**

**Sprite.AmmoKit.hide()**

### Algorithm to get the top scores of the player.

The database is arranged to record data like this:

Username, score1, score2, score3, score4, score5, totalCash, sensitivity setting, bar/number for ammo setting, bar/number for HP setting, etc the finally \n, it is multidimensional array. Passwords are saved in a hashing table.

At login, the code will save the users data into an object called userInfo, this file can be accessed easily to get data from it like settings, scores, and usernames; it will also save the line it was on into the variable called dataLine. The algorithm will read the scores, then sort them in order the display them at the end of each game and see if they must be edited, or from the menu, once it has done that it will sort them and then reinsert it into the list called file and into the database.txt. It will also delete the smallest one from the list so that there are never more than 5 scores.

**PlayerHighScore(score, file, dataLine):** #if it is being called from the menu then the score will be zero. File is the data that is pulled during login about the user.

**if score == 0:**

**displayedScore = ‘’**

**else**

**displayedScore = ‘Your score is,’ score**

**for i in 0 to 4:**

**scores[i] = file[2+i]**

**scores.append(score)**

**for i in range(0, length(scores))**

**key = scores[i]**

**k = i-1**

**while k>= 0 and key < scores[k]:**

**scores[i] = scores[i-1]**

**k =- 1**

**scores[k+1] = key**

**pop(scores)**

**print(“Your scores”)**

**for i in 0 to 4:**

**print(“score”, i,”. ”,scores[i])**

**print(displayedScore)**

**totalCash =+ cash**

**file[2] = scores[0]**

**file[3] = scores[1]**

**file[4] = scores[2]**

**file[5] = scores[3]**

**file[6] = scores[4]**

**file[7] = totalCash**

**datafile = open(‘database.txt’, ‘read’)**

**datafile.close()**

**datafile.split(‘\n’)**

**datafile[dataLine] = file**

**files = open(‘database.txt’, ‘write’)**

**files.write(datafile)**

**close()**

**print(“All players scores scores”)**

**allScores = []**

**for i in len(datafile):**

**for j in 0 to 4:**

**allScores.append(datafile[i][2+j]**

**for i in range(0, length(allScores))**

**key = allScores[i]**

**k = i-1**

**while k>= 0 and key < allScores[k]:**

**allScores[i] = allScores[i-1]**

**k =- 1**

**allScores[k+1] = key**

**top5allScores = []**

**for i in range(0,4):**

**top5allScores.append(allScores[i]**

**print(“Leader board scores”)**

**for i in 0 to 4:**

**print(“score”, i,”. ”, top5allScores[i])**

**print(‘Coins earned in game’, cash)**

### Databases clean up.

It will open the database and see if there are any gaps and then remove them and then reprint them into the txt file. It reads the database, then splits everything line by line, then finds when there are gaps, and then pops them and then writes them into the txt file.

**Function Cleanup():**

**files = open("Database.txt","r")**

**data = files.readlines()**

**files.close()**

**counter = len(data)-1**

**for i in range(0,len(data)):**

**temp = data[counter].rsplit("\n")**

**temper = str(temp)**

**if temper == "":**

**data.pop(counter)**

**counter = counter - 1**

**files = open("Database.txt", "w")**

**for i in range(0,len(users)):**

**temper = str(users[i])**

**files.write(temper)**

**files.close()**

### Login

The code will get a username and a password, it will then check the username against the database and sees if it is there, if it is not, it then returns false; if it, does it save the account details to the global variable file and the account line in datafile.

**Function Login:**

**inputtedUsername = input(“Username”)**

**files = open(‘Database.txt’,‘read’)**

**details = files.readlines()**

**for i in range(0, len(details):**

**if inputtedUsername == details[i][0]**

**there = True**

**file = detail[i]**

**dataLine = i**

**break()**

**else()**

**there = False**

**if there == False:**

**return False**

**else:**

**inputtedPassword= input(“Password”)**

**address = searchPass(inputtedUsername, inputtedPassword)**

**if address == False**

**return False**

**else:**

**return True**

### Password verification

It takes the inputted username and then the inputted password, then it hashes the username and password into a hash table, it returns the line number where the password is and false if it is not there, if the inputted password is ‘9342976867’ then it will return the next open spot in the table with the hashed username, this is for account creation purposes.

**def searchPass(user, upass):**

**files = open("Password.txt","r") # copies list to a list locally**

**words = files.readlines()**

**files.close()**

**userWlist = list(user)**

**num = 0**

**for i in range(0,len(userWlist)):**

**num = num + ord(userWlist[i])**

**address = num % 33 # got address**

**gnum = hashlib.sha512(upass.encode())**

**hash = gnum.hexdigest()**

**Lopo = True**

**count = 0**

**while Lopo == True:**

**if address >= 32:**

**address = 0**

**count = count + 1**

**if words[address].rstrip("\n")== hash and words[address].rstrip("\n") != "del":**

**return address**

**if words[address].rstrip("\n") == "del":**

**if upass == 9342976867:**

**return address**

**else:**

**address = address+1**

**if words[address].rstrip("\n") == "":**

**if upass == 9342976867:**

**return address**

**else:**

**return False**

**if words[address].rstrip("\n") != "" and words[address].rstrip("\n")!=hash:**

**address = address+1**

**if count > 12:**

**return False**

### Account Deletion

It first requests for the username and password for confirmation then it searches the usernames to see if it is correct, then it hashes the name and the password to find it in the password files, it reads it to see if it is there but if it is not, then the function fails; but if it succeeds it will return if it has done it. If it returns True, it means the account has been deleted and the game goes back to the login, if it returns False, 0 means that there are only 2 accounts, so no deleting more. If it returns False,1 means that the usernames do not match up, if it is False,3 then the passwords do not match up.

**def removeLogin(file, dataLine):**

**files = open("Database.txt","r") # copies list to a list locally**

**users = files.readlines()**

**files.close()**

**if len(users)<=2:**

**print("There is only 2 username can't delete last ones")**

**return (False, 0)**

**there = False**

**WtR = input("Write your username: ")**

**if WtR =! file[0]:**

**print("Incorrect username of player")**

**return (False, 1)**

**else:**

**there = True**

**upass = input("Input your password: ")**

**if there == True:**

**address = searchPass(WtR,upass)**

**if address == False:**

**print("incorrect password, retry")**

**return (False,2)**

**else:**

**files = open("Passwords.txt","r")**

**words = files.readlines()**

**files.close()**

**words[address] = "del\n"**

**files = open("Passwords.txt", "w")**

**for i in range(0,len(words)):**

**temp = str(words[i])**

**files.write(temp)**

**files.close()**

**files = open("Database.txt","r")**

**account = files.readlines()**

**files.close()**

**accounts[dataLine] = "\n"**

**files = open("Datbase.txt", "w")**

**for i in range(0,len(users\_rL)):**

**temp = str(users\_rL[i])**

**files.write(temp)**

**files.close()**

**Cleanup():**

**ruturn True**

### Keeping user data

The user data will be used frequently during the game, so best to keep it in an object of a class. This allows it to be stored and got but without accidentally changing the user’s data.

**Class user:**

**Private username**

**Private cash**

**Private score1**

**Private score2**

**Private score3**

**Private score4**

**Private score5**

**Private rotation\_method**

**Private rotation\_sensitivity**

**Private maxHP**

**Private maxAmmo**

**Private regenHP**

**Private regenAmmo**

**Private volume**

**Private fireType**

**Private hudtype**

**Private colour**

**Private hook**

**Public procedure new(data):**

**Username = data[0]**

**cash= data[1]**

**score1= data[2]**

**score2= data[3]**

**score3= data[4]**

**score4= data[5]**

**score5= data[6]**

**rotation\_method= data[7]**

**rotation\_sensitivity= data[8]**

**maxHP= data[9]**

**maxAmmo= data[10]**

**regenHP= data[11]**

**regenAmmo= data[12]**

**volume= data[13]**

**fireType= data[14]**

**hudtype= data[15]**

**colour= data[16]**

**hook= data[17]**

**Public procedure getName():**

**return username**

**Public procedure getCash():**

**return int(cash)**

**Public procedure setCash(new):**

**cash = new**

**Public procedure getMaxAmmo():**

**return int(maxAmmo)**

**Public procedure setMaxAmmo (new):**

**maxAmmo = new**

**Public procedure getMaxHP():**

**return int(maxHP)**

**Public procedure setMaxHP(new):**

**maxHP = new**

**Public procedure getRegenHP():**

**return int(regenHP)**

**Public procedure setRegenHP(new):**

**regenHP = new**

**Public procedure getRegenAmmo():**

**return int(regenAmmo)**

**Public procedure setkitAmmo(new):**

**regenAmmo = new**

**Public procedure getSens():**

**return int(rotation\_sensitivity)**

**Public procedure setSens(new):**

**rotation\_sensitivity = new**

**Public procedure getVolume():**

**return int(volume)**

**Public procedure setVolume(new):**

**volume = new**

**Public procedure getS1():**

**return s1**

**Public procedure setS1(new):**

**s1 = new**

**Public procedure getS2():**

**return s2**

**Public procedure setS2(new):**

**s2 = new**

**Public procedure getS3():**

**return s3**

**Public procedure setS3(new):**

**s3 = new**

**Public procedure getS4():**

**return s4**

**Public procedure setS4(new):**

**s4 = new**

**Public procedure getS5():**

**return s5**

**Public procedure setS5(new):**

**s5 = new**

**Public procedure getHud():**

**return int(Hudtype)**

**Public procedure setHud(new):**

**hudtype = new**

**Public procedure getRotate():**

**return int(rotation\_method)**

**Public procedure setRotate(new):**

**rotation\_method = new**

**Public procedure getFire():**

**return int(fireType)**

**Public procedure setFire(new):**

**fireType = new**

**Public procedure getColour():**

**return int(colour)**

**Public procedure setColour(new):**

**colour = new**

**Public procedure getHook():**

**return int(hook)**

**Public procedure setHook(new):**

**hook = new**

## Tokens

|  |  |
| --- | --- |
| Name | What it holds, Why |
| file | During login the user’s data is logged into a global list so it can be accessed everywhere. |
| score | The score that was achieved in a game. A variable. |
| Online | Started as soon as the games booted up, so stuff knows it must loop. Boolean. |
| GameLoop | So, stuff that must loop can loop. Boolean. |
| HP | Health points a player has during a game loop. Variable integer. |
| Ammo | Ammo points a player has during a game loop. Variable integer. |
| usernameInputed | The inputted username by the player |
| passwordInputed | The inputted password by the player |
| ticks | A global variable that increases in the game loop, speeding up enemies’ movement. |
| collisionCheck() | Checks if the parameter is an enemy |
| random.randint(1,20)) | Gives a random number between the 2 numbers in it. |
| Ammo\_multiplier | Has a base number of 1 and can be upgraded in the shop that increases the ammo each ammo kit will give. |
| totalCash | Is a global variable assigned during login from the database containing users’ total cash |
| Cash | It is a variable assigned to each new game that gives the total cash given per game. |
| MaxX, MaxY | The variable contains the size of the screen of the game window, which helps find the middle. |
| searchPass(WtR,upass) | Takes the username and hashes it to find the location, then hashes the password and sees if it matches the address, if it is, then it returns the line number, and if it is not then it returns false. |
| Delay() | Delays on the integer in the brackets, seconds. |
| userInfo | It is the object to be instanced in the class user to store the user data. |

## Data storage structures

|  |  |
| --- | --- |
| What | Why |
| Class system (called user) | The storage of data of the user through the game. |
| Database (Called userData.txt) | The user data will be stored in a table that links to the hash table by using the username. |
| Hash table (called hashTable) | To store the passwords in a secure way so that the password is secure. |
| Records (used to store data in database) | How data will be stored in the table. It is a logical way of showing data so easier to understand. |
| 2-dimensional array. (called allScores) | When showing the score, the person sees the username and score. I sort the score and not the username. It needs to be 2-dimensional as it eases use when displaying it. |
| Singleton Variable (called Globals) | How variables from different scripts are accessed throughout the game. |
| Local and global variables | Need it to run a program. |
| Lists | This is how I will store a record in the database. |

## Testing method

Testing of the finished game can be done in stages:

|  |  |
| --- | --- |
| Stage number | What is tested |
| 1 | The login and account creating mechanic |
| 2 | The menu, setting editor and the shop |
| 3 | The game loop |
| 4 | The after-game record keeping |

The testing of the game will be done by me and people who I know and will happen during programming and post-development to make sure there are no bugs. This will be black box testing.

The prototypes will be testing the bits that have changed and will be fully white box.

The reason the data is needed is to allow further iterations or game patches or new game balances to be rolled out with changes. The testing data is used to find out any improvements to be made or changes to be made to make the game better.

### Stage 1

#### Account making

For the account making, I will attempt to make an account; the data inputted could be letters, numbers, and symbols. Once inputted I shall check the database, in it I expect a line with my username and the 5 zeros marking as the user has not played a game, so has no score, once done I check the hash table txt containing passwords and see if a new password is there. There will be more rounds of this until I reach the max number of accounts possible to be made (10) and see if it allows me to make more.

#### Login

The login testing will be 5 phases long, all the inputted data will be either a letter, number, or a symbol.

##### Phase 1

Phase 1 is where I try to use a faulty username but a correct password. I expect that it will report an incorrect username/password.

##### Phase 2

Phase 2 is where I use a correct username but a faulty password. I expect that it will report an incorrect username/password.

##### Phase 3

Phase 3 is where both username and password are faulty. I expect that it will report an incorrect username/password.

##### Phase 4

Phase 4 is where both correct username and password are used but for different accounts. I expect that it will report an incorrect username/password.

##### Phase 5

Phase 5 is where I see if it will allow entry into an account if both the username and password are correct. I expect that it will lead me into the menu.

### Stage 2

#### Menu

The menu allows access into the game, into the setting editor, into showing the high scores of all current users, the score of the user, into the shop and closing the game. The tests for this part will be just clicking the option and seeing if they will lead into what was designated.

#### Settings editor

The settings editor allows the user to change a multitude of settings (music, movement sensitivity, sound effects, delete the account, reset player, key bindings, if you want a bar or a number for ammo and HP display) once those are done the user presses save, this will update the user’s database records. Once those settings are saved, I will check the database and see if it really has changed and not just a variable. The inputted data will either be a click of the mouse or sliding a bar for sensitivity on the designated setting.

|  |  |
| --- | --- |
| Setting | Input method for testing |
| Bar or number for ammo/HP | Click |
| Sensitivity | Sliding a bar |
| Sound effects | Click |
| Music | Click |
| Reset | Click |
| Delete account | Click |
| Keybind changing | Click |

Once the settings are altered, if the player presses the save button it will save the changes to the database, if they did not it will only change the local list on the player’s data, this will only affect the current game and not the future games. Once done they can exit the settings editor, once done it will lead into the menu.

#### Shop

This allows upgrading of the character, from increasing the health pool to getting more health back from a tank-repair-kit. This test will see if the shop works, by increasing the recorded upgrades in their account details. The only input will be the clinking of the mouse as that will either decrease the totalCash and then update the player’s settings on their upgrade or will either fail as they have insufficient funds, or it will report the item cannot be upgraded anymore. I expect that the database will have also changed for the parts that record upgrades.

The next test is to see if I am led back to the menu when I press the close button.

### Stage 3

#### Game loop

The game loop testing stage will be the longest and hardest as we will be testing all the enemies, the playable character, the tank-repair-kits, and ammo kits and the sound system. The code editing for it will also be rigorous.

There will be 8 parts for this section.

##### Phase 1 – movement

Checking if the sprites can move. This involves seeing if the player rotates when the correct keybind is pressed, seeing if the enemies move towards the player, and if the ammo and tank-repair kits appear and disappear properly. I expect that the turret will rotate, the enemy to move towards the player and that the kits will appear randomly on the screen.

##### Phase 2 – shells

The test will include testing:

|  |  |
| --- | --- |
| Testing | Expected |
| Pressing the correct keybind to launch the shell | The shell will move in the direction of the turret |
| Seeing what happens when the shell touches the enemy | They should change their shape into an explosion and then cause all the enemies that touch it to hide and then increase the variable score and then then the variable cash. |
| What happens to the shell when it reaches the boundary? | It will delete that instance of the shell |
| What happens if the key bind of the shell is kept pressed | There will be a delay of 0.5 seconds per launch |

##### Phase 3 – hook

The hook is launched when its specific keybind is pressed, it is used to get the kits. The tests of it will include:

|  |  |
| --- | --- |
| Testing | Expected |
| What happens when the keybind is pressed? | The hook will launch from the turret. |
| What happens if the hook touches the tank-repair-kit | The hook and kit will disappear increasing the health by the base health increase plus the upgraded number. If the max health is smaller than the increase than the health, then it will increase the health to the HP limit. |
| What happens if the hook touches the ammo kit | The hook and kit will disappear increasing the ammo by the base health increase plus the upgraded number. |
| What happens if the hook reaches the game boundary? | It will delete that instance of the hook. |
| What happens if the key bind of the hook is kept pressed | There will be a delay of 2 seconds per launch |

##### Phase 4 – enemy

The test for this is to see if all the enemies work correctly, that is disappearing if touched by the shell, increasing the score, increasing the cash, disappearing if it touches the turret, and decreasing the HP. They all must face the turret and move towards it with a velocity that increases linearly throughout the gameplay by the time multiplier. The test is to see: if the enemy’s velocity increases if the game goes on long enough; if the enemy faces the player; if the enemy increases the score and cash when they die when they touch the turret shell; see if the enemy disappears when it touched the turret, decreasing the HP.

##### Phase 5– kits

The kits will have very similar code; the only difference is what they look like, what the increase and whet multiplier they have. The test is to see if the kits will appear, the next test is to see if they disappear when touching the hook, increasing their receptive variable when disappearing, the next test is to see if they randomly appear after a delay of 5-30 seconds.

### Stage 4

This will happen after the gameplay and will show your totalCash, cash you earned in the game, the score, the score compared to your other 5 recorded scores, and your score compared to the top 5 scores of all the other users.

The test is to see if the data is accurate.

## Further tests that are done post-development

Further tests can be done post-development to weed out bugs and improve efficiency in the code. These will be black box and white box testing.

### White box

This is where I will test the code while looking at the code, while going through the game. I will probably use studio editor as it allows step running. This could be testing how user-friendly the game is or how intuitive it is. The data gathered would be any bug finds or any game balancing fixes I can find, that can be used in later iterations to improve the game.

### Black Box

This is where I will test the game without checking the code; this will be done by my beta testers. The data gathered would be any bug finds or any game balancing fixes I can find. The beta testers would also tell me of further developments that could be patched on later game releases.

# Software development

## Prototype 1

Prototype 1 will be all the game aspects done like settings editor, shop, scores tab, and the game itself.

### Day 09/05/2021

This is when I started coding, my aim for the day was to finish of the registration and the base for all the instances to load onto.

#### Main v1

|  |  |
| --- | --- |
| What is it? | This is the base from which I will launch most scenes like the game window or the shop or the menu.  This is done by using 2 singleton variable that tell me what scene to open and if the window is empty so I can reload a scene. |
| Troubles | * I had no idea on how to launch an instance but when I asked my teacher. * I had no idea on how to make singleton variable until I watched a tutorial on how to use the autoloader to make them. |
| Script | extends Node2D  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  var a = 2  #launchs the login scene  func \_ready():      if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      pass # Replace with function body.  #checks the global variables to see if it needs to launch something  func \_process(\_delta):      if Global.rege == 1:          if Global.logopen == false:              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0:          if Global.logopen == false:              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      pass |
| Scene | It is empty as a blank canvas for all the other scenes |

#### Login v1

|  |  |
| --- | --- |
| What is it? | This scene is loaded onto Main when it is opened then it allows you to press a button to change the Global scene decider variable to the one for the registration scene and the scene open variable to false. Once done that itself terminates so does not overlay the registration scene. |
| Troubles | This is not complex enough to have troubles. |
| Script | extends Control  func \_ready():      pass  func \_on\_Reg\_pressed():      Global.rege = 1      Global.logopen = false      self.queue\_free() |
| Scene |  |

#### Registration v1

|  |  |
| --- | --- |
| What is it? | This scene is loaded from the Main if the user presses the registration button in the Login. Once loaded it checks if the max accounts have reached, if there is then it hides all the nodes expect the button to go back to the Login scene. Otherwise, the user can input a username and password then confirm it. The user then presses the register button which then sees if the username is already in use, if it is then launches the wrong(), otherwise it the see if the passwords match in the boxes. Then sees if the password is valid, if all is true it then uses that data and launches the addLogin(user, passw) function.  The addLogin function first copies all the user data in the userData.txt then it appends to the string using concatenation the username and the standard file data. It then hashes the password then uses the Global function hashedLocation to find the location in the list of hashed passwords in hashPass.txt it must replace, once that is done it save the data to the file then uses the singleton function cleanup().  If the function cleanup is used, then it shows Label2 the hides it after 2 seconds |
| Troubles | * I did not know how to make a timer in Godot, but when I looked in the forum, I found I could use “yield(get\_tree().create\_timer(2.0), "timeout")” to make a one-shot timer of a specified time to hide Label2 again. * I did not know how to access and write to files in Godot but when I researched it, I found it is like python. |
| Script | |
| extends Control    #files in userData saved in format of username, normal\_sensitivity(1), shellMax(20), HPMax(10),  #hpKitsize(2),ammokitSize(10),highscore1,highscore2,highscore3,highscore4,highscore5\n  #makes the labels that show if your accepted and if you details are invalid  #checks the files to see if there is already the max amount of accounts  #if there is then it hides the text boxes and only shows the back and Label  func \_ready():      $Label1.hide()      $Label2.hide()      var file = File.new()      var count = 0      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var arrcontent = content.split("\n")      for n in len(arrcontent):          count += 1        if count >= 11:          $Label1.show()          $Reg.hide()          $Userbox.hide()          $Passbox.hide()          $ConPassbox.hide()      pass  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return false          if  userline[0] == "":              return true          if  userline[0] == " ":              return true      return true  func passCheck(passw): # see if given invalid username      if passw == "del":          return false      if passw == " ":          return false      if passw == "":          return false      return true    func addLogin(user, passw): # add details to text files      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var file2 = File.new()      file2.open("res://userData.txt", file.WRITE)      var add = content+"\n"+user + ",1,20,10,2,10,0,0,0,0,0"      file2.store\_line(add)      file2.close()      var loco = Global.hashedLocation(user)      var hashed = passw.sha256\_text()      var file3 = File.new()      file3.open("res://HashPass.txt", file.READ)      var content2 = file3.get\_as\_text()      content2 = content2.split("\n", true)      print(content2)      file3.close()      var file4 = File.new()      file4.open("res://HashPass.txt", file.WRITE)      content2[loco] = hashed      for i in len(content2):          file4.store\_line(str(content2[i]))      file4.close()  func wrong(): # shows the label2 if the details are invalid      $Label2.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $Label2.hide()  func \_on\_Reg\_pressed(): # is a signal receiver from the button Reg      var user = $Userbox.text      var passw = $Passbox.text      var findu = searchUser(user)      if $Passbox.text == $ConPassbox.text:          if findu == false:              wrong()              pass          if findu == true:              var findp = passCheck(passw)              if findp == false:                  wrong()                  pass              else:                  addLogin(user,passw)                  Global.cleanup()      else:          wrong()          pass    func \_on\_Back\_pressed():# is a signal receiver from the button Back leads to the Login scene      Global.rege = 0      Global.logopen = false      self.queue\_free() | |
| Scene |  |

#### globals v1

|  |  |
| --- | --- |
| What is it? | It is an autoloaded script that had all singleton variables and the cleanup and hashedLocation function.  Cleanup reads the userData.txt file then turns it into a string by splitting at “\n” then remove all gaps or parts of the string the has gaps the saves the file.  hashedLocation function is used in registration to find where the password is saved in the hash table. Id does this by turning the username into a list then uses the inbuilt ord function to get the denary of that value. It then divides that number until it is below 20. Then checks the file to see if there is a gap at that location and loops until it finds one. |
| Troubles | I did not know how to get the denary value of a character without using a large function in GDscript until I looked it up and found the ord function. |
| Script | extends Node    var logopen = false  var rege = 0  var logged = false  func cleanup(): # cleans up the file      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()    func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 20:          num = num /19      print(num)      var count = 0      var loop = true      while loop == true:          if num >= 19:              num = 0          count =+ 1          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| To see if adding an account works | It will add the login details to the file | N/A | It worked.   |  | | --- | |  | |  | |
| To see if the registration button in the Login scene works | It will load up the registration scene and delete the login scene | N/A | Works   |  | | --- | |  | |  | |
| To see if the check for max account works. Test1 | It will hide all text boxes and show the label that says maxed account reached | N/A | Did not work |
| To see if the check for max account works. Test2 | It will hide all text boxes and show the label that says maxed account reached | Change the if to >= and make the nodes hide and show with .hide() and .show() than using $node. visible =!visible | It worked. |
| Test all buttons I have coded for to make sure they work | All buttons do as they have been programmed | N/A | They did as they were intended to do. |

#### Summary

On this iteration I coded for the registration and storing of user data into a file. This partially completes the requirement of having a password system.

### Day 10/05/2021

This day I planned to code for the Login. The only scripts that were edited were globals and Login compared to their previous versions.

#### Login v2

|  |  |
| --- | --- |
| What is it? | This will lead to the registration and will also read the text boxes and see if you got the correct login details if you do, I plan to code in the menu next. If the login is false, it shows the Label2 |
| Troubles | Most of the code was reused from the globals script. |
| Script | extends Control  func \_ready():      $Label1.hide()      $Label2.hide()      pass  func \_on\_Log\_pressed(): # checks to see if that username exists the tells us if it is false      var user = $Userbox.text#or not, then tries to find the hashed password in hash table      var find = searchUser(user)# if it does i plan to make it open the menu next      if find == false:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 1          Global.logopen = false          self.queue\_free()      var passw = $Passbox.text      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              Global.logged = true              break          elif content[num] == "":              break          elif content[num] == " ":              break          elif content[num] == "end":              num = 0          if count > 10:              return false          num = num +1      if Global.logged == true:          $Label1.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 1          Global.logopen = false          self.queue\_free()      else:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 0          Global.logopen = false          self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              Global.userData = userline              return true          if  userline[0] == "":              return false          if  userline[0] == " ":              return false      return true  func \_on\_Reg\_pressed(): # opens registration scene      Global.rege = 1      Global.logopen = false      self.queue\_free() |
| Scene |  |

#### globals v2

|  |  |
| --- | --- |
| What is it? | Same as last version but more variable and changed hashedLocation. |
| Troubles | Same as last version |
| Script | extends Node  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var logopen = false  var rege = 0  var logged = false  var userData = []  # Called when the node enters the scene tree for the first time.  func cleanup():      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()    func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      print(num)      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1    # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| See if it will login works. | Will see the registration scene | N/A | It worked. |

#### Summary

On this iteration I coded for the login system, this partially fulfils the success criteria for having a strong password system.

### Day 11/05/2021

For this day I plan to finish off the menu with the shop, scores viewer. Of the previous versions of the script only the Login, Main and globals are changed.

#### Main v2

|  |  |
| --- | --- |
| What is it? | Same as last time but more stuff is being instanced from here |
| Troubles | Just a repetition of what I did already. |
| Script | extends Node2D  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  var a = 2  func \_ready():      Global.cleanup()      if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      pass  func \_process(\_delta):      if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true      if Global.rege == 4: # rege 3 = scores          if Global.logopen == false:              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true      pass |
| Scene | It is empty as a blank canvas for all the other scenes |

#### Login v3

|  |  |
| --- | --- |
| What is it? | Same as previous version but the code will launch the menu instead of the registration. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  # Called when the node enters the scene tree for the first time.  func \_ready():      $Label2.hide()      $Label1.hide()      pass # Replace with function body.  func \_on\_Log\_pressed():      var user = $Userbox.text      var find = searchUser(user)      if find == false:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 1          Global.logopen = false          self.queue\_free()      var passw = $Passbox.text      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              Global.logged = true              break          elif content[num] == "":              break          elif content[num] == " ":              break          elif content[num] == "end":              num = 0          if count > 10:              return false          num = num +1      if Global.logged == true:          $Label1.show()          yield(get\_tree().create\_timer(0.5), "timeout")          Global.rege = 2          Global.logopen = false          self.queue\_free()      else:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 0          Global.logopen = false          self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              Global.userData = userline              return true          if  userline[0] == "":              return false          if  userline[0] == " ":              return false      return true  func \_on\_Reg\_pressed():      Global.rege = 1      Global.logopen = false      self.queue\_free()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |
| Scene |  |

#### globals v3

|  |  |
| --- | --- |
| What is it? | Same as last version but more variable. |
| Troubles | Same as last version |
| Script | extends Node  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var logopen = false  var rege = 0  var logged = false  var userData = []  var shopOpen = false  var newHP = 0  var newHP\_cost = int(newHP\*1.5)  var newAmmo = 0  var newAmmo\_cost = int(newAmmo\*1.5)  var newMed = 0  var newMed\_cost = int(newMed\*2)  var newAmmok = 0  var newAmmok\_cost = int(newAmmok\*1.5)  # Called when the node enters the scene tree for the first time.  func cleanup():      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()    func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      print(num)      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1    # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |

#### Menu v1

|  |  |
| --- | --- |
| What is it? | It is a location where buttons are used, and their response changes the scene variable and can show your cash and allow you to quit |
| Troubles | I did not know to quit the game but when I looked it up it told me get tree then use the inbuilt function .quit() |
| Script | extends Control    func \_ready():#reads variable to show your cash      $Label1.hide()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))    func \_on\_Quit\_pressed(): # closes game      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      get\_tree().quit()  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func \_on\_Shop\_pressed(): # opens shop and closes menu      Global.rege = 3      Global.logopen = false      self.queue\_free()  func \_on\_Scores\_pressed(): # planned to work      pass |
| Scene |  |

#### Shop v1

|  |  |
| --- | --- |
| What is it? | Allows to upgrade the users’ stats like max health, ammo, recovered hp and ammo from kits. |
| Troubles |  |
| Script | |
| extends Control  func \_ready(): # get the variable filled in      $NotEnough.hide()      Global.newHP = int(Global.userData[3])+2      Global.newHP\_cost = int(Global.newHP\*1.5)      Global.newAmmo = int(Global.userData[2])+3      Global.newAmmo\_cost = int(Global.newAmmo\*1.5)      Global.newMed = int(Global.userData[4])+1      Global.newMed\_cost = int(Global.newMed\*1.5)      Global.newAmmok = int(Global.userData[5])+7      Global.newAmmok\_cost = int(Global.newAmmok\*1.5)    func \_process(\_delta): # keeps the text live        $Money.text = "Total Cash:" + str(int(Global.userData[1]))      var hptext = "Currently:" + Global.userData[3] +", upgraded:"+ str(Global.newHP)+ ", upgarde cost:" + str(Global.newHP\_cost)      $Hp.text = str(hptext)      var ammotext = "Upgrade Max Ammo, Currently:" + Global.userData[2] + ", upgraded:"+ str(Global.newAmmo)+ ", upgrade cost:" + str(Global.newAmmo\_cost)      $Ammo.text = str(ammotext)      var med =  "Upgrade Medkit regen ammount, Currently:" + Global.userData[4] +", upgraded:"+str(Global.newMed)+",upgrade cost:" + str(Global.newMed\_cost)      $Med.text = str(med)      var kit = "Upgrade Ammokit regen ammount, Currently:" + Global.userData[5] + ", upgraded:" + str(Global.newAmmok) +",upgrade cost:" + str(Global.newAmmok\_cost)      $Kit.text = str(kit)    #2345  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func change(cost): # saves the changes      Global.userData[1] = str(int(Global.userData[1])-cost)      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = Global.userData[0]+","+Global.userData[1]+","+Global.userData[2]+","+Global.userData[3]+","+Global.userData[4]+","+Global.userData[5]+","+Global.userData[6]+","+Global.userData[7]+","+Global.userData[8]+","+Global.userData[9]+","+Global.userData[10]+","+Global.userData[11]      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == Global.userData[0]:                content[i] = replacing          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      Global.cleanup()      pass  func check(cost): # see if they can afford it      var newMoney = int(Global.userData[1])-cost      if newMoney >= 0:          return true      else:          return false  func notEnough(): # if they cannot afford item      print("hi3")      $NotEnough.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $NotEnough.hide()      pass  func \_on\_AmmoMax\_pressed(): # upgrades max ammo      var enoughCash = check(Global.newAmmo\_cost)      if enoughCash == true:          Global.userData[2] = str(int(Global.userData[2]) +5)          change(Global.newAmmo\_cost)          Global.newAmmo = int(Global.userData[3])+5          Global.newAmmo\_cost = int(Global.newAmmo\*1.5)      else:          notEnough()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))      var ammotext = "Upgrade Max Ammo, Currently:" + Global.userData[2] + ", upgraded:"+ str(Global.newAmmo)+ ", upgrade cost:" + str(Global.newAmmo\_cost)      $Ammo.text = str(ammotext)        pass  func \_on\_Ammokit\_pressed():# upgrades max ammo regen from kit      var enoughCash = check(Global.newAmmok\_cost)      if enoughCash == true:          Global.userData[5] = str(int(Global.userData[5])+3)          change(Global.newAmmok\_cost)          Global.newAmmok = int(Global.userData[3])+5          Global.newAmmok\_cost = int(Global.newAmmok\*1.5)      else:          notEnough()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))      pass  func \_on\_HP\_pressed(): # upgrades max hp      var enoughCash = check(Global.newHP\_cost)      if enoughCash == true:          Global.userData[3] = str(int(Global.userData[3]) +5)          change(Global.newHP\_cost)          Global.newHP = int(Global.userData[4])+2          Global.newHP\_cost = int(Global.newHP\*1.5)      else:          notEnough()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))      var hptext = "Currently:" + Global.userData[3] +", upgraded:"+ str(Global.newHP)+ ", upgarde cost:" + str(Global.newHP\_cost)      $Hp.text = str(hptext)      pass  func \_on\_MedKit\_pressed():# upgrades max hp regen from kit      print("hi")      var enoughCash = check(Global.newMed\_cost)      if enoughCash == true:          print("hi")          Global.userData[4] = str(int(Global.userData[4]) +5)          change(Global.newMed\_cost)          Global.newHP = int(Global.userData[4])+1          Global.newHP\_cost = int(Global.newHP\*2)      else:          notEnough()          print("hi2")      $Money.text = "Total Cash:" + str(int(Global.userData[1]))      var med =  "Upgrade Medkit regen ammount, Currently:" + Global.userData[4] +", upgraded:"+str(Global.newMed)+",upgrade cost:" + str(Global.newMed\_cost)      $Med.text = str(med)      pass    func \_on\_Menu\_pressed():#loads up the menu      Global.rege = 2      Global.logopen = false      self.queue\_free()      pass # Replace with function body. | |
| Scene |  |

#### Scores v1

|  |  |
| --- | --- |
| What is it? | Allow the user to see their top 5 scores and the top scores on all accounts. |
| Troubles | Did not know how to sort a 2-column array. But I looked at some old code of mine for a coding project and try and sort it. |
| Script | |
| extends Control  var allScores =[]  # loads in scores in order  func \_ready():      $PersonalScores.text = "1."+Global.userData[7]+"\n"+"2."+Global.userData[8]+"\n"+"3."+Global.userData[9]+"\n"+"4."+Global.userData[10]+"\n"+"5."+Global.userData[11]+"\n"      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      users.remove(len(users)-1)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          for j in 5:              name = userline[0]+"."              var add = []              add.append(name)              print(userline[7+j])              add.append(int(userline[7+j]))              add.append("\n")              print("i")              allScores.append(add)        for i in len(allScores):          var key = allScores[i][1]          var j = i-1          while j >=0:              while key < allScores[j][1]:                  allScores[j+1] = allScores[j]                  j -= 1                  print("j")              allScores[j+1] = key      allScores = allScores.invert()      allScores = str(allScores)      allScores.replace("[","")      allScores.replace("]","")      print(allScores)  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free() | |
| Scene |  |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| See if it will display cash | Shows cash |  |  |
| See if the login loaded in the menu | Will see the menu scene | N/A | It worked. |
| See if open shop button works | Will see shop | N/A |  |
| See if shop works | See if upgrades work |  | Did not change. |
| See if shop works | See if upgrades work | Put $NotEnough.hide() in the ready not the process and then changed the mistake of using newHP\_cost in the medkit function | Worked |

#### Summary

The shop scene allows the upgrading of the character fulfilling the success criteria of upgrading of a player’s character. The scores scene allows the player to see their scores and the other players partially fulfilling the criteria for having a score system as it is not tested yet.

### Day 12/05/2021

On this day I planned to do the settings and start on the game by making the Game and HUD. I also edited the Main, Menu, Scores and globals.

#### Main v3

|  |  |
| --- | --- |
| What is it? | Same as last time but more stuff is being instanced from here |
| Troubles | Just a repetition of what I did already. |
| Script | extends Node2D  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  func \_ready():      Global.cleanup()      if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      pass  func \_process(\_delta):      if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true      if Global.rege == 4: # rege 4 = scores          if Global.logopen == false:              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true      if Global.rege == 5: # rege 5 = settings          if Global.logopen == false:              var setting = Settings.instance()              var main = get\_tree().current\_scene              main.add\_child(setting)              Global.logopen = true      if Global.rege == 6: # rege 6 = account deletion          if Global.logopen == false:              var delete = Delete.instance()              var main = get\_tree().current\_scene              main.add\_child(delete)              Global.logopen = true      if Global.rege == 7: # rege 7= starts game          if Global.logopen == false:              var game = Game.instance()              var main = get\_tree().current\_scene              main.add\_child(game)              Global.logopen = true |
| Scene | It is empty as a blank canvas for all the other scenes |

#### globals v4

|  |  |
| --- | --- |
| What is it? | Same as last version but more variable. |
| Troubles | Same as last version |
| Script | extends Node  var logopen = false  var rege = 0  var logged = false  var userData = []  var shopOpen = false  var pause = 0 # if 1 means pause game play, if 0 means let gameplay continue  var game = 0 # 0 means no game, 1 means currently playing  var hp = 0  var ammo = 0  var cash = 0  var score = 0  var newHP = 0  var newHP\_cost = int(newHP\*1.5)  var newAmmo = 0  var newAmmo\_cost = int(newAmmo\*1.5)  var newMed = 0  var newMed\_cost = int(newMed\*2)  var newAmmok = 0  var newAmmok\_cost = int(newAmmok\*1.5)  # Called when the node enters the scene tree for the first time.  func cleanup():      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()    func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      print(num)      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1 |

#### Menu v2

|  |  |
| --- | --- |
| What is it? | Same as last version but more scenes are linked to it. |
| Troubles |  |
| Script | extends Control    func \_ready():      $Label1.hide()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))    func \_on\_Quit\_pressed():      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      get\_tree().quit()  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func \_on\_Shop\_pressed():      Global.rege = 3      Global.logopen = false      self.queue\_free()  func \_on\_Game\_pressed():      Global.rege = 7      Global.logopen = false      self.queue\_free()  func \_on\_Settings\_pressed():      Global.rege = 5      Global.logopen = false      self.queue\_free()  func \_on\_Scores\_pressed():      Global.rege = 4      Global.logopen = false      self.queue\_free()      pass |
| Scene |  |

#### Scores v2

|  |  |
| --- | --- |
| What is it? | Same as last time but got the bubble sort to work |
| Troubles | Used some old code that was in python to solve it |
| Script | extends Control  var allScores =[]  #loads in scores in order, personal and local  func \_ready():      $PersonalScores.text = "1. "+Global.userData[8]+"\n"+"2. "+Global.userData[9]+"\n"+"3. "+Global.userData[10]+"\n"+"4. "+Global.userData[11]+"\n"+"5. "+Global.userData[12]+"\n"      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      print(content)      for i in len(content)-1:          var line = content[i]          line = line.split(",", true)          print(line)          var add1 = line[0]+". "+line[8]          var add2 = line[0]+". "+line[9]          var add3 = line[0]+". "+line[10]          var add4 = line[0]+". "+line[11]          var add5 = line[0]+". "+line[12]          allScores.append(add1)          allScores.append(add2)          allScores.append(add3)          allScores.append(add4)          allScores.append(add5)      for i in range(allScores.size()-1, -1, -1):          for j in range(1,i+1,1):              var notes = allScores[j].split(". ", true)              var prevnotes = allScores[j-1].split(". ", true)              if notes[1] > prevnotes[1]:                  var temp = allScores[j-1]                  allScores[j-1] = allScores[j]                  allScores[j] = temp      var scoresOrdered = ""      for i in len(allScores):          scoresOrdered = scoresOrdered + allScores[i] + "\n"      $HighScores.text = scoresOrdered  #sends back to menu  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free() |
| Scene |  |

#### Settings v1

|  |  |
| --- | --- |
| What is it? | Let’s you change the turrets sensitivity, if sound will be on or off, which HUD type you want, if you want to reset your account and if you want to delete your account. Most are just reactionary to a signal when an interactive is used. All are done in code but the delete which uses a separate scene. |
| Troubles | At first, I did not know how to use the drop-down boxes, but I watched a tutorial and learnt how to. |
| Script | |
| extends Control  func \_ready(): # adds stuff to the drop-down box      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")  func \_process(\_delta):      $Sens.value = int(Global.userData[6])      $Label.text = "Sensitivity of Turret, Currently: " + str(Global.userData[6])      if Global.userData[7] == "0":          $Sound.text = "Sound is currently: Off, Press to switch"      else:          $Sound.text = "Sound is currently: On, Press to switch"    func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userData[6] = value    func \_on\_Sound\_pressed():      if Global.userData[7] == "0":          Global.userData[7] = "1"      else:          Global.userData[7] = "0"  func \_on\_Delete\_pressed():      Global.rege = 6      Global.logopen = false      self.queue\_free()  func \_on\_Reset\_pressed():      Global.userData[1] = "0"      Global.userData[2]="20"      Global.userData[3]="10"      Global.userData[4]="2"      Global.userData[5]="10"      Global.userData[6]="1"      Global.userData[7]="1"      Global.userData[8]="0"      Global.userData[9]="0"      Global.userData[10]="0"      Global.userData[11]="0"      Global.userData[12]="0"      Global.userData[13]="2"      \_on\_Menu\_pressed()  func \_on\_HUD\_item\_selected(type):      print(type)      Global.userData[13] = str(type)  func \_on\_Menu\_pressed():      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = Global.userData[0]+","+Global.userData[1]+","+Global.userData[2]+","+Global.userData[3]+","+Global.userData[4]+","+Global.userData[5]+","+Global.userData[6]+","+Global.userData[7]+","+Global.userData[8]+","+Global.userData[9]+","+Global.userData[10]+","+Global.userData[11]+","+Global.userData[12]+","+Global.userData[13]      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == Global.userData[0]:                content[i] = replacing          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      Global.cleanup()      Global.rege = 2      Global.logopen = false      self.queue\_free()      print("hi")      pass # Replace with function body. | |
| Scene |  |

#### Delete v1

|  |  |
| --- | --- |
| What is it? | This is launched from Main when the user presses the Delete node in settings, this is used to delete the user’s account. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  func \_ready():      $Label1.hide()  func \_on\_Settings\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return true      return false  func incorrectLogin():      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $Label1.hide()      $Userbox.text = "Username"      $Passbox.text = "Password"  func \_on\_Delete\_pressed():      var userName = $Userbox.text      var passWord = $Passbox.text      var foundUser = searchUser(userName)  #gets username and password and sees if it can delete it      if foundUser == false:          incorrectLogin()          pass      else:          var foundPass = realPass(userName, passWord)          if foundPass == -1:              incorrectLogin()              pass          else:              Global.userData = []              var file = File.new()              file.open("res://userData.txt", file.READ)              var content = file.get\_as\_text()              content = content.split("\n",true)              file.close()              var counter = len(content)-1              for i in len(content):                  var splitContent = content[i].split(",",true)                  if splitContent[0] == userName:                        content[i] = ""                  counter = counter - 1              file.open("res://userData.txt", file.WRITE)              for g in len(content):                  file.store\_line(content[g])              file.close()              Global.cleanup()              file.open("res://HashPass.txt", file.READ)              content = []              content = file.get\_as\_text().split("\n",true)              print(content)              file.close()              content[foundPass] = "del"              file.open("res://HashPass.txt", file.WRITE)              for i in len(content)-1:                  file.store\_line(str(content[i]))              file.close()              Global.rege = 0              Global.logopen = false              self.queue\_free()      func realPass(user ,passw):      var hashed = passw.sha256\_text()  #returns the location of the password in the array      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()        var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              return  num          elif content[num] == "":              return -1          elif content[num] == " ":              return -1          elif content[num] == "end":              num = 0          if count > 10:              return -1          num = num +1  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |
| Scene |  |

#### Game v1

|  |  |
| --- | --- |
| What is it? | It is the base layer on which the game loop lies on, it is supposed to instance near everything in the game loop. It also allows the game to pause and reset. |
| Troubles |  |
| Script | extends Node2D  const HUD = preload("res://Scenes/HUD.tscn")  func \_ready():      Global.game = 1      var HUD = HUD.instance()      var main = get\_tree().current\_scene      main.add\_child(HUD)  func \_process(\_delta):      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      if Global.pause == 1:          pass |
| Scene | It is empty as a blank canvas for all the game sprites |

#### HUD v1

|  |  |
| --- | --- |
| What is it? | Tells us our max health, ammo current ammo, health and our cash and score in a text format of bar, the settings allow you to decide if you want one specific one or both. It also allows you to pause the game |
| Troubles |  |
| Script | extends Control  var Hudtype = int(Global.userData[13])  func \_ready(): # gets everything ready      $ProHUDHP.set\_max(int(Global.userData[3]))      $ProHUDAmmo.set\_max(int(Global.userData[2]))      Global.hp = int(Global.userData[3])      Global.ammo = int(Global.userData[2])      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      pass    #allow you to pause  func \_on\_Pause\_pressed():      $Pause.hide()      Global.pause = 1      $Unpause.show()      $Quit.show()      #unpause  func \_on\_Unpause\_pressed():      $Pause.show()      Global.pause = 0      $Unpause.hide()      $Quit.hide()  #allows you to quit  func \_on\_Quit\_pressed():      Global.game = 0      self.queue\_free()  #the process that keeps the HUD accurate  func \_process(\_delta):      $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      if Hudtype == 0:          $HUD.text = "HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.hide()          $ProHUDAmmo.hide()      elif Hudtype == 1:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      else:          $HUD.text = "HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo) |
| Scene |  |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| See if the deletion of account works | It will delete the user’s account. | N/A | It worked.   |  | | --- | |  | |  | |
| See if the setting editor works to change HUD type | The HUD will change |  | Did not work crashed |
| See if the setting editor works to change HUD type take 2 | The HUD will change | Change the error of ProHUdHP to ProHUDHP | |  | | --- | |  | |  | |  | |
| See if the leader board work | Show scores in order from greatest to lowest |  | Works |
| See if account reset works | Account details set to default |  | Works   |  | | --- | |  | |  | |
| See if you can exit the game loop with the HUD | The main scene is shown |  | It happened |

#### Summary

The settings scene allows a player to change sensitivity of the player, reset player data partially fulfilling the games criteria and control the volume however as now sound currently plays it is only partially successful. The delete scene allows a user to delete their accounts, this fulfils its task by deleting the user data from the database and their password from the hash table fulfilling its success criteria. The score viewer has now been testes but hasn’t been tested for different scores than 0, partially succeeding in its intended purpose. The HUD has been made but the changing of values has not been tested so it partially fulfils its success criteria.

### Day 13/05/2021

On this day I wanted to code in the player and the part of the shell. The only scripts that are different are Settings, HUD, Game, globals and registration.

#### Game v2

|  |  |
| --- | --- |
| What is it? | Same as last time but also making the global variables that are used in the game loop are reset. |
| Troubles |  |
| Script | extends Node2D  const HUD = preload("res://Scenes/HUD.tscn")  const Player = preload("res://Scenes/Player.tscn")  func \_ready():  #reset the variables      Global.pause = 0      Global.game = 1      Global.newGame = 0  #opens the player      var player = Player.instance()      var main2 = get\_tree().current\_scene      main2.add\_child(player)  #opens the HUD      var HUD = HUD.instance()      var main = get\_tree().current\_scene      main.add\_child(HUD)  func \_process(\_delta):      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      if Global.pause == 1:          pass      if Global.newGame == 1:          Global.logopen = false          self.queue\_free() |
| Scene | It is empty as a blank canvas for all the game sprites |

#### globals v5

|  |  |  |
| --- | --- | --- |
| What is it? | Because of the many times I had to write code to save changes I made a global function to save all data for me. And more variables. | |
| Troubles | Same as last version | |
| Script | |
| extends Node  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var logopen = false  var rege = 0  var logged = false  var userData = []  var shopOpen = false  var pause = 0 # if 1 means pause game play, if 0 means let gameplay continue  var game = 0 # 0 means no game, 1 means currently playing  var hp = 1  var ammo = 1  var cash = 0  var score = 0  var newGame = 0  var firing = 0  var newHP = 0  var newHP\_cost = int(newHP\*1.5)  var newAmmo = 0  var newAmmo\_cost = int(newAmmo\*1.5)  var newMed = 0  var newMed\_cost = int(newMed\*2)  var newAmmok = 0  var newAmmok\_cost = int(newAmmok\*1.5)  # Called when the node enters the scene tree for the first time.  func cleanup():      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()    func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      print(num)      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1    func save\_changes():      Global.userData[1] = str(int(Global.userData[1])+Global.cash)      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = Global.userData[0]+","+Global.userData[1]+","+Global.userData[2]+","+Global.userData[3]+","+Global.userData[4]+","+Global.userData[5]+","+Global.userData[6]+","+Global.userData[7]+","+Global.userData[8]+","+Global.userData[9]+","+Global.userData[10]+","+Global.userData[11]+","+Global.userData[12]+","+Global.userData[13]+","+Global.userData[14]+","+Global.userData[15]      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == Global.userData[0]:                content[i] = replacing          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      cleanup()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass | |

#### HUD v2

|  |  |
| --- | --- |
| What is it? | Same as last time but programmed in the bit that stops the game when health is 0, it also shows the top scores and current score in a list |
| Troubles |  |
| Script | extends Control  var Hudtype = int(Global.userData[13])  var scores = []  #reset some variables and hide some labels  func \_ready():      $ProHUDHP.set\_max(int(Global.userData[3]))      $ProHUDAmmo.set\_max(int(Global.userData[2]))      Global.hp = int(Global.userData[3])      Global.ammo = int(Global.userData[2])      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      $Over.hide()  #get the list ready for sorting      scores.append(int(Global.userData[8]))      scores.append(int(Global.userData[9]))      scores.append(int(Global.userData[10]))      scores.append(int(Global.userData[11]))      scores.append(int(Global.userData[12]))      pass    #pauses game  func \_on\_Pause\_pressed():      $Pause.hide()      Global.pause = 1      $Unpause.show()      $Quit.show()  #unpauses game  func \_on\_Unpause\_pressed():      $Pause.show()      Global.pause = 0      $Unpause.hide()      $Quit.hide()  #makes the game loop end  func \_on\_Quit\_pressed():      Global.game = 0      self.queue\_free()  #keeps the scores live  func \_process(\_delta):      $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      if Hudtype == 0:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\nAmmo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.hide()          $ProHUDAmmo.hide()      elif Hudtype == 1:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      else:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      if Global.hp == 0:          if Global.pause == 0:              Global.pause = 1              $Pause.hide()              changes()  #goes back to menu  func \_on\_Menu\_pressed():      Global.game = 0      self.queue\_free()  #restart game loop  func \_on\_Restart\_pressed():      Global.newGame = 1      self.queue\_free()  #makes the text box show your scores in order  func changes():      var change = false      scores.append(Global.score)        for i in range(scores.size()-1, -1, -1):          for j in range(1,i+1,1):              var notes = scores[j]              var prevnotes = scores[j-1]              if notes > prevnotes:                  var temp = scores[j-1]                  scores[j-1] = scores[j]                  scores[j] = temp                  change = true      for i in len(scores):          $Scores.text = $Scores.text + str(scores[i]) + "\n"      if change == true:          Global.userData[8]= str(scores[0])          Global.userData[9]= str(scores[1])          Global.userData[10]=str(scores[2])          Global.userData[11]=str(scores[3])          Global.userData[12]=str(scores[4])      Global.save\_changes() |
| Scene |  |

#### Player v1

|  |  |
| --- | --- |
| What is it? | This is the sprite the player is controlling; it rotates only by the command of the assigned key in settings. It also is where the shell is launched from |
| Troubles | I had no idea on how to make the player move until I read the Godot wiki on 2d movement and saw how you use the rotation+= rotation\_dir\* rotation\_speed \*delta. Rotation speed is the base speed of 1 multiplied by the sensitivity decided in the settings. I also did not know to make the shell launch from the tank and how to keep it rotated until my teacher told me about instancing and I have used that to get the shell to launch from the tank. |
| Script | extends KinematicBody2D  const Shell = preload("res://Scenes/Shell.tscn")  export (float) var rotation\_speed = 5  var sensitivity = int(Global.userData[6])  var rotation\_dir = 0  var left = ""  var right = ""  var fire = ""  # gets the correct signals for the player  func \_ready():      if int(Global.userData[14]) == 0:          left = "ui\_right"          right = "ui\_left"      else:          right = "ui\_a"          left = "ui\_d"      if int(Global.userData[15]) == 2:          fire = "ui\_r"      elif int(Global.userData[15]) == 1:          fire = "ui\_space"      else :          fire = "ui\_f"      pass # Replace with function body.  #gets the player moving  func \_process(delta):      rotation\_speed = 5 \* sensitivity      rotation\_dir = 0      if Global.game == 0:          self.queue\_free()      if Global.pause == 1:          pass      elif Input.is\_action\_pressed(left):          rotation\_dir += 1      elif Input.is\_action\_pressed(right):          rotation\_dir -= 1      elif Input.is\_action\_pressed(fire):          if Global.firing == 0:              if Global.ammo > 0:                  var shell = Shell.instance()                  var main = get\_tree().current\_scene                  main.add\_child(shell)                  shell.global\_transform = $Muzzle.global\_transform                  Global.firing = 1                  Global.ammo = Global.ammo - 1      rotation += rotation\_dir \* rotation\_speed \* delta |
| Scene |  |

#### Registration v2

|  |  |
| --- | --- |
| What is it? | This is the same as last time but as there are more setting to save the registration must be edited. They are the setting to the way you shoot and the way you rotate. |
| Troubles |  |
| Script | |
| extends Control    #files in userData saved in format of username,cash,shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10),normal\_sensitivity(1),sound(1=true, 0 =false)  #,highscore1,highscore2,highscore3,highscore4,highscore5,type of HUD 0 = only text 1 = progress bar 2 = both, how to rotate 0 = left and right arrow 1 = a d keys, how to fire 0 = f 1 =space 2 = r \n  func \_ready():      $Label1.hide()      $Label2.hide()      Global.cleanup()      var file = File.new()      var count = 0      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var arrcontent = content.split("\n")      for n in len(arrcontent):          count += 1        if count >D= 11:          $Label1.show()          $Reg.hide()          $Userbox.hide()          $Passbox.hide()          $ConPassbox.hide()      pass # Replace with function body.    func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return false          if  userline[0] == "":              return true          if  userline[0] == " ":              return true      return true  func passCheck(passw):      if passw == "del":          return false      if passw == " ":          return false      if passw == "":          return false      return true    func addLogin(user, passw): #adds stuff to the txt files about user      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      file.close()      var file2 = File.new()      file2.open("res://userData.txt", file.WRITE)      var add = content+"\n"+user +",0,20,10,2,10,1,1,0,0,0,0,0,2,1,1"      file2.store\_line(add)      file2.close()      var loco = Global.hashedLocation(user)      var hashed = passw.sha256\_text()      var file3 = File.new()      file3.open("res://HashPass.txt", file.READ)      var content2 = file3.get\_as\_text()      content2 = content2.split("\n", true)        file3.close()      var file4 = File.new()      file4.open("res://HashPass.txt", file.WRITE)      content2[loco] = hashed      for i in len(content2)-1:          file4.store\_line(str(content2[i]))      file4.close()  func wrong():      $Label2.visible = visible      yield(get\_tree().create\_timer(2.0), "timeout")      $Label2.visible = !visible  func \_on\_Reg\_pressed():      var user = $Userbox.text      var passw = $Passbox.text      var findu = searchUser(user)      if $Passbox.text == $ConPassbox.text:          if findu == false:              wrong()              pass          if findu == true:              var findp = passCheck(passw)              if findp == false:                  wrong()                  pass              else:                  addLogin(user,passw)                  Global.cleanup()                  Global.rege = 0                  Global.logopen = false                  self.queue\_free()      else:          wrong()          pass      func \_on\_Back\_pressed():      Global.rege = 0      Global.logopen = false      self.queue\_free() | |
| Scene |  |

#### Settings v2

|  |  |
| --- | --- |
| What is it? | The same as last time but more stuff as more settings can be edited as they need to be able to change the way they rotate and shoot. Also used the global function to save the users data. |
| Troubles | At first, I did not know how to use the drop-down boxes, but I watched a tutorial and learnt how to. |
| Script | extends Control  func \_ready():      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")      $Movement.add\_item("Left and Right arrow keys")      $Movement.add\_item("Using the A and D keys")        $Fire.add\_item("The F key")      $Fire.add\_item("The Space bar")      $Fire.add\_item("The R key")  func \_process(\_delta):      $Sens.value = int(Global.userData[6])      $Label.text = "Sensitivity of Turret, Currently: " + str(Global.userData[6])      if Global.userData[7] == "0":          $Sound.text = "Sound is currently: Off, Press to switch"      else:          $Sound.text = "Sound is currently: On, Press to switch"    func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userData[6] = value    func \_on\_Sound\_pressed():      if Global.userData[7] == "0":          Global.userData[7] = "1"      else:          Global.userData[7] = "0"  func \_on\_Delete\_pressed():      Global.rege = 6      Global.logopen = false      self.queue\_free()  func \_on\_Reset\_pressed():      Global.userData[1] = "0"      Global.userData[2]="20"      Global.userData[3]="10"      Global.userData[4]="2"      Global.userData[5]="10"      Global.userData[6]="1"      Global.userData[7]="1"      Global.userData[8]="0"      Global.userData[9]="0"      Global.userData[10]="0"      Global.userData[11]="0"      Global.userData[12]="0"      Global.userData[13]="2"      Global.userData[14]="1"      Global.userData[15]="1"      \_on\_Menu\_pressed()  func \_on\_HUD\_item\_selected(type):      print(type)      Global.userData[13] = str(type)  func \_on\_Menu\_pressed():      Global.save\_changes()      Global.rege = 2      Global.logopen = false      self.queue\_free()      print("hi")      pass # Replace with function body.  func \_on\_Movement\_item\_selected(type):      print(type) # how to rotate 0 = left and right arrow 1 = a d keys      Global.userData[14] = str(type)  func \_on\_Fire\_item\_selected(type):      print(type)#how to fire 0 = f 1 =space 2 = r      Global.userData[15] = str(type) |
| Scene |  |

#### Shell v1

|  |  |
| --- | --- |
| What is it? | It will move from the shell in the direction the tank is when the fire key was pressed. It will continue until it is out of the screen then it will self-terminate. |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -500  var timer  var hit = 0  func \_physics\_process(delta):      position += transform.y \* speed \* delta    func ready():      yield(get\_tree().create\_timer(0.1), "timeout")      Global.firing = 0  func \_on\_Out\_screen\_exited():      Global.firing = 0      self.queue\_free() |
| Scene |  |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| See if the tank will rotate | It will rotate in the direct I pressed | N/A | It worked.   |  | | --- | |  | |  | |
| Will the shell fire if the fire key is pressed | It will fire the shell |  | Worked |

#### Summary

This iteration made the player and allows it to rotate to the set keybind (that can be changed in the updated settings scene) so partially fulfils criteria of rotating the player. The shell made moves, but it has not been coded for collisions, so it only slightly fulfils its intended task. The HUD has been updated to stop the game at 0 health now.

### Day 14/05/2021

On this day I coded in the zombies, and I edited the globals, Shell and Game.

#### Game v3

|  |  |
| --- | --- |
| What is it? | Same as last time but also a way to spawn zombies and increase the tick speed. |
| Troubles |  |
| Script | extends Node2D  const HUD = preload("res://Scenes/HUD.tscn")  const Player = preload("res://Scenes/Player.tscn")  const Zombie = preload("res://Scenes/Zombie.tscn")  var launch = 0  export (float) var time = 0  #so that there is a wait between spawns of zombies  func wait():      yield(get\_tree().create\_timer(1.5), "timeout")      launch = 0  func \_ready():      Global.pause = 0      Global.game = 1      Global.newGame = 0      Global.ticks = 0      var player = Player.instance()      var main2 = get\_tree().current\_scene      main2.add\_child(player)      var HUD = HUD.instance()      var main = get\_tree().current\_scene      main.add\_child(HUD)      var loop = true      while loop == true:          yield(get\_tree().create\_timer(2.0), "timeout")          Global.ticks = Global.ticks + 0.5  func \_process(\_delta):      print(Global.ticks)      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      if Global.pause == 1:          pass        if Global.newGame == 1:          Global.logopen = false          self.queue\_free()      if Global.game == 1:          if Global.pause == 0:              if launch == 0:                  #launches zombies                  if Global.zombieNum < 20:                      var zombie = Zombie.instance()                      var main = get\_tree().current\_scene                      main.add\_child(zombie)                      Global.zombieNum =Global.zombieNum  +  1                      launch = 1                      wait() |
| Scene | It is empty as a blank canvas for all the game sprites |

#### globals v6

|  |  |
| --- | --- |
| What is it? | Same as last time but more variables |
| Troubles | Same as last version |
| Script | |
| extends Node  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var logopen = false  var rege = 0  var logged = false  var userData = []  var shopOpen = false  var pause = 0 # if 1 means pause game play, if 0 means let gameplay continue  var game = 0 # 0 means no game, 1 means currently playing  var hp = 1  var ammo = 1  var cash = 0  var score = 0  var newGame = 0  var firing = 0  var zombieNum = 0  export (float) var ticks = 0  var newHP = 0  var newHP\_cost = int(newHP\*1.5)  var newAmmo = 0  var newAmmo\_cost = int(newAmmo\*1.5)  var newMed = 0  var newMed\_cost = int(newMed\*2)  var newAmmok = 0  var newAmmok\_cost = int(newAmmok\*1.5)  # Called when the node enters the scene tree for the first time.  func cleanup():      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()    func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.txt", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 33:          num = num /33      print(num)      var count = 0      var loop = true      while loop == true:          if num >= 33:              num = 0          count = count + 1          print(count)          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1    func save\_changes():      Global.userData[1] = str(int(Global.userData[1])+Global.cash)      var file = File.new()      file.open("res://userData.txt", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = Global.userData[0]+","+Global.userData[1]+","+Global.userData[2]+","+Global.userData[3]+","+Global.userData[4]+","+Global.userData[5]+","+Global.userData[6]+","+Global.userData[7]+","+Global.userData[8]+","+Global.userData[9]+","+Global.userData[10]+","+Global.userData[11]+","+Global.userData[12]+","+Global.userData[13]+","+Global.userData[14]+","+Global.userData[15]      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == Global.userData[0]:                content[i] = replacing          counter = counter - 1        file.open("res://userData.txt", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      cleanup()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass | |

#### Shell v2

|  |  |
| --- | --- |
| What is it? | Same as last time but collision detection. I also coded it to show an animation when it explodes |
| Troubles | I did not know how to make collision detection so looked it up on the wiki and used the move\_and\_collide function and then got collision data from it that is used to find the name of the collider with collision.collider.name then programmed it to play the animation and increase the collision hit box to 3 instead of 2. |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = 0  var playing = 0  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.hide()      $Sprite.hide()      yield(get\_tree().create\_timer(0.1), "timeout")      Global.firing = 0  func \_physics\_process(delta):      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      if Global.pause == 1:          pass      velocity += transform.y \* speed      if hit == 0:          collision = move\_and\_collide(velocity \* delta)      if hit == 0:          if collision:              if str(collision.collider.name) == "Zombie":                  hit = 1      if hit == 1:          if playing == 0:                $Bullet.hide()              $Sprite.show()              $CollisionShape2D.scale.x = 3              $CollisionShape2D.scale.y = 3              $Sprite.frame = 0              $Sprite.playing = true              playing = 1  func \_on\_Sprite\_animation\_finished():        Global.firing = 0      self.queue\_free()  func \_on\_Out\_screen\_exited():      Global.firing = 0      self.queue\_free() |
| Scene |  |

#### Zombie v1

|  |  |
| --- | --- |
| What is it? | It will randomly move to one of the sides of the screen then look toward the tank then start moving, if hit by the shell they increase the score and cash but if they hit the tank, they reduce health by one. Using an animated sprite to animate it. |
| Troubles | I did not know how to code for random numbers but then I looked it up and found how to use the RandomNumberGenerator by making a variable with it with “.new()” and then use the “.randomize()” to change the seed then use “randi\_range(a,b)” to get random integers |
| Script | extends KinematicBody2D  export (int) var speed = 100  var velocity = Vector2()  var collision  var hit = 0  var rng = RandomNumberGenerator.new()  #decides the place of starting for zombie  func \_ready():      rng.randomize()      $Sprite.playing= true      var where = rng.randi\_range(0,1) # 0 means on base 1 = on height      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              var along =  rng.randi\_range(0,600)              position.y = along            else:              position.x = 1024              var along =  rng.randi\_range(0,600)              position.y = along        else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              var along =  rng.randi\_range(0,1024)              position.x = along          else:              position.y = 600              var along =  rng.randi\_range(0,1024)              position.x = along  #makes it move and disappear if hit  #if hit give score and cash  func \_physics\_process(delta):      var target = Vector2(512, 400)      look\_at(target)      velocity = position.direction\_to(target)        if position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)        if collision:          if str(collision.collider.name) == "Shell":              hit = 1          if str(collision.collider.name) == "Tank":              Global.hp = Global.hp -1              self.queue\_free()      if hit == 1:          Global.score = Global.score +1 + int(Global.ticks)          Global.cash = Global.cash +1 + int(Global.ticks)          Global.zombieNum = Global.zombieNum-1          self.queue\_free()      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      if Global.pause == 1:          $Sprite.playing= false          pass |
| Scene |  |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| See if the zombies move towards the tank | Zombies spawn randomly and move towards the tank | N/A | Worked |
| If the pause button will stop the zombies | Zombies stop |  | Did not work |
| If the pause button will stop the zombies | Zombies stop | Put the move\_and\_collide into an else statement under the pause check | Worked |

#### Summary

The shell has been updated to detect collisions so fully fulfils its success criteria. The zombie scene moves and collides with the tank and the shells so fulfils its success criteria as it reduces the players health. The HUD has also been tested and it allows the player to pause the game.

### Day 15/05/2021

On this day I overhauled my code by changing from using a txt to dat file to store stuff then used 32 as the max index in Godot. I also implemented the use of the mouse to control the tank (mouse position is rotation and when the user clicks it is a fire). I also implemented the hook and both the ammo and med kit. I also implemented a new scene to the menu for the user to see their controls.

#### Main v4

|  |  |
| --- | --- |
| What is it? | Same as last time but more scenes but also a way to load up the hashPass.dat file if it is not there. |
| Troubles | Just a repetition of what I did already. |
| Script | extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  # Declare member variables here. Examples:  # var a = 2  # var b = "text"    # Called when the node enters the scene tree for the first time.  func \_ready():      Global.cleanup()        if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      print(doFileExists)      var remake = []      if doFileExists == false:          for i in 34:              print(i)              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()      pass # Replace with function body.        func \_process(\_delta):      if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true      if Global.rege == 4: # rege 4 = scores          if Global.logopen == false:              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true      if Global.rege == 5: # rege 5 = settings          if Global.logopen == false:              var setting = Settings.instance()              var main = get\_tree().current\_scene              main.add\_child(setting)              Global.logopen = true      if Global.rege == 6: # rege 6 = account deletion          if Global.logopen == false:              var delete = Delete.instance()              var main = get\_tree().current\_scene              main.add\_child(delete)              Global.logopen = true      if Global.rege == 7: # rege 7= starts game          if Global.logopen == false:              var game = Game.instance()              var main = get\_tree().current\_scene              main.add\_child(game)              Global.logopen = true      if Global.rege == 8: # rege 8= starts controls          if Global.logopen == false:              var controls = Controls.instance()              var main = get\_tree().current\_scene              main.add\_child(controls)              Global.logopen = true      pass |
| Scene | It is empty as a blank canvas for all the other scenes |

#### Login v4

|  |  |
| --- | --- |
| What is it? | Change the use of txt file to .dat file then use an index of 32 than 34 |
| Troubles |  |
| Script | extends Control  func \_ready():      $Label2.hide()      $Label1.hide()      pass  func \_on\_Log\_pressed():  #checks to see if login is correct  #if yes it move on if not then  #it will show the Label2  #which tell us that the login is incorrect      var user = $Userbox.text      var find = searchUser(user)      if find == false:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 1          Global.logopen = false          self.queue\_free()      var passw = $Passbox.text      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()        var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              Global.logged = true              break          elif content[num] == "":              break          elif content[num] == " ":              break          elif content[num] == "end":              num = 0          if count > 10:              return false          num = num +1      if Global.logged == true:          $Label1.show()          yield(get\_tree().create\_timer(0.5), "timeout")          Global.rege = 2          Global.logopen = false          self.queue\_free()      else:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 0          Global.logopen = false          self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              Global.userData = userline              return true          if  userline[0] == "":              return false          if  userline[0] == " ":              return false      return true  func \_on\_Reg\_pressed():      Global.rege = 1      Global.logopen = false      self.queue\_free()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |
| Scene |  |

#### Registration v3

|  |  |
| --- | --- |
| What is it? | Same as last time but use userData.dat and hashPass.dat than userData.txt and hashPass.txt |
| Troubles |  |
| Script | |
| extends Control    #files in userData saved in format of username,cash,shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10),normal\_sensitivity(1),sound(1=true, 0 =false)  #,highscore1,highscore2,highscore3,highscore4,highscore5,type of HUD 1 = only text 2 = progress bar 3 = both, how to rotate 1 = left and right arrow 2 = a d keys 3 =mouse position,  #how to fire 1 = f 2 =space 2 = r 3= click\n  func \_ready():      $Label1.hide()      $Label2.hide()      Global.cleanup()      var file = File.new()      var count = 0      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var arrcontent = content.split("\n")      for n in len(arrcontent):          count += 1        if count >= 11:          $Label1.show()          $Reg.hide()          $Userbox.hide()          $Passbox.hide()          $ConPassbox.hide()      pass # Replace with function body.    func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return false          if  userline[0] == "":              return true          if  userline[0] == " ":              return true      return true  func passCheck(passw):      if passw == "del":          return false      if passw == " ":          return false      if passw == "":          return false      if passw == "end":          return false      return true    func addLogin(user, passw):      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var file2 = File.new()      file2.open("res://userData.dat", file.WRITE)      var add = content+"\n"+user +",0,20,10,2,10,1,1,0,0,0,0,0,3,1,1"      file2.store\_line(add)      file2.close()      var loco = Global.hashedLocation(user)      var hashed = passw.sha256\_text()      var file3 = File.new()      file3.open("res://HashPass.dat", file.READ)      var content2 = file3.get\_as\_text()      content2 = content2.split("\n", true)        file3.close()      var file4 = File.new()      file4.open("res://HashPass.dat", file.WRITE)      content2[loco] = hashed      for i in len(content2)-1:          file4.store\_line(str(content2[i]))      file4.close()  func wrong():      $Label2.visible = visible      yield(get\_tree().create\_timer(2.0), "timeout")      $Label2.visible = !visible  func \_on\_Reg\_pressed():      var user = $Userbox.text      var passw = $Passbox.text      var findu = searchUser(user)      if $Passbox.text == $ConPassbox.text:          if findu == false:              wrong()              pass          if findu == true:              var findp = passCheck(passw)              if findp == false:                  wrong()                  pass              else:                  addLogin(user,passw)                  Global.cleanup()                  Global.rege = 0                  Global.logopen = false                  self.queue\_free()      else:          wrong()          pass      func \_on\_Back\_pressed():      Global.rege = 0      Global.logopen = false      self.queue\_free() | |
| Scene |  |

#### globals v7

|  |  |
| --- | --- |
| What is it? | Same as last time but more variables and change from .txt to .dat |
| Troubles | Same as last version |
| Script | |
| extends Node  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var logopen = false  var rege = 0  var logged = false  var userData = []  var shopOpen = false  var pause = 0 # if 1 means pause game play, if 0 means let gameplay continue  var game = 0 # 0 means no game, 1 means currently playing  var hp = 1  var ammo = 1  var cash = 0  var score = 0  var newGame = 0  var firing = 0  var zombieNum = 0  export (float) var ticks = 0  var hook = 0  var mKit = 0  var aKit = 0    # Called when the node enters the scene tree for the first time.  func cleanup():      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()              func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)        file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32        var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1            if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1    func save\_changes():      Global.userData[1] = str(int(Global.userData[1])+Global.cash)      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = Global.userData[0]+","+Global.userData[1]+","+Global.userData[2]+","+Global.userData[3]+","+Global.userData[4]+","+Global.userData[5]+","+Global.userData[6]+","+Global.userData[7]+","+Global.userData[8]+","+Global.userData[9]+","+Global.userData[10]+","+Global.userData[11]+","+Global.userData[12]+","+Global.userData[13]+","+Global.userData[14]+","+Global.userData[15]      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == Global.userData[0]:                content[i] = replacing          counter = counter - 1        file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      cleanup()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass | |

#### Menu v3

|  |  |
| --- | --- |
| What is it? | Same as last version but a scene that tells the user the controls. |
| Troubles |  |
| Script | extends Control    func \_ready():      $Label1.hide()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))    func \_on\_Quit\_pressed():      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      get\_tree().quit()  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func \_on\_Shop\_pressed():      Global.rege = 3      Global.logopen = false      self.queue\_free()  func \_on\_Game\_pressed():      Global.rege = 7      Global.logopen = false      self.queue\_free()  func \_on\_Settings\_pressed():      Global.rege = 5      Global.logopen = false      self.queue\_free()  func \_on\_Controls\_pressed():      Global.rege = 8      Global.logopen = false      self.queue\_free()  func \_on\_Scores\_pressed():      Global.rege = 4      Global.logopen = false      self.queue\_free()      pass # Replace with function body. |
| Scene |  |

#### Controls v1

|  |  |
| --- | --- |
| What is it? | This scene will allow the user to see the controls of the user. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var movement = ""  var shootS = ""  var hook = "When the H key is pressed"  # Called when the node enters the scene tree for the first time.  func \_ready():      if int(Global.userData[14]) == 1:          movement = "When the Left and Right Arrow Keys are pressed"      elif int(Global.userData[14]) == 2:          movement = "When the A and D keys are pressed"      else:          movement = "Using the global mouse position"      if int(Global.userData[15]) == 3:          shootS = "When the R key is pressed"      elif int(Global.userData[15]) == 2:          shootS = "When the Space bar is pressed"      elif int(Global.userData[15]) == 1 :          shootS = "When the F key is pressed"      else:          shootS = "When the mouse is clicked"      $Stuff.text = " Movement of player is:"+ movement+ "\n How to shoot the Shell: "+ shootS + "\n How fire the hook: " + hook      pass # Replace with function body.  func \_on\_Menu\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free()      pass |
| Scene |  |

#### Scores v3

|  |  |
| --- | --- |
| What is it? | Same as last time but got the bubble sort to work |
| Troubles | Used some old code that was in python to solve it |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var allScores =[]  # Called when the node enters the scene tree for the first time.  func \_ready():      $PersonalScores.text = "1. "+Global.userData[8]+"\n"+"2. "+Global.userData[9]+"\n"+"3. "+Global.userData[10]+"\n"+"4. "+Global.userData[11]+"\n"+"5. "+Global.userData[12]+"\n"      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      print(content)      for i in len(content)-1:          var line = content[i]          line = line.split(",", true)          print(line)          var add1 = line[0]+". "+line[8]          var add2 = line[0]+". "+line[9]          var add3 = line[0]+". "+line[10]          var add4 = line[0]+". "+line[11]          var add5 = line[0]+". "+line[12]          allScores.append(add1)          allScores.append(add2)          allScores.append(add3)          allScores.append(add4)          allScores.append(add5)        for i in range(allScores.size()-1, -1, -1):          for j in range(1,i+1,1):              var notes = allScores[j].split(". ", true)              var prevnotes = allScores[j-1].split(". ", true)              if int(notes[1]) > int(prevnotes[1]):                  var temp = allScores[j-1]                  allScores[j-1] = allScores[j]                  allScores[j] = temp      print(allScores)      var scoresOrdered = ""      for i in len(allScores):          scoresOrdered = scoresOrdered + allScores[i] + "\n"      $HighScores.text = scoresOrdered  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free() |
| Scene |  |

#### Settings v3

|  |  |
| --- | --- |
| What is it? | Same as last time but the display is different, and I used to disable boxes to show what the boxes options are. I also put the Label under Sens so it will be attached. I also added more options like using the mouse to rotate or fire. The factory reset also will needed tweaking as the options all moved up 1. |
| Troubles |  |
| Script | extends Control  func \_ready():      $HUD.add\_item("Change on how you see the HUD in Game")      $HUD.set\_item\_disabled(0, true)      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")      $Movement.add\_item("Change on how you rotate player")      $Movement.set\_item\_disabled(0, true)      $Movement.add\_item("Left and Right arrow keys")      $Movement.add\_item("Using the A and D keys")      $Movement.add\_item("Using the Mouse location")      $Fire.add\_item("Change on how you fire")      $Fire.set\_item\_disabled(0, true)      $Fire.add\_item("The F key")      $Fire.add\_item("The Space bar")      $Fire.add\_item("The R key")      $Fire.add\_item("Using a mouse click")  func \_process(\_delta):      $Sens.value = int(Global.userData[6])      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userData[6])      if Global.userData[7] == "0":          $Sound.text = "Sound is currently: Off, Press to switch"      else:          $Sound.text = "Sound is currently: On, Press to switch"    func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userData[6] = value    func \_on\_Sound\_pressed():      if Global.userData[7] == "0":          Global.userData[7] = "1"      else:          Global.userData[7] = "0"  func \_on\_Delete\_pressed():      Global.rege = 6      Global.logopen = false      self.queue\_free()  func \_on\_Reset\_pressed():      Global.userData[1] ="0"      Global.userData[2]="20"      Global.userData[3]="10"      Global.userData[4]="2"      Global.userData[5]="10"      Global.userData[6]="1"      Global.userData[7]="1"      Global.userData[8]="0"      Global.userData[9]="0"      Global.userData[10]="0"      Global.userData[11]="0"      Global.userData[12]="0"      Global.userData[13]="3"      Global.userData[14]="1"      Global.userData[15]="1"      \_on\_Menu\_pressed()  func \_on\_HUD\_item\_selected(type):        Global.userData[13] = str(type)  func \_on\_Menu\_pressed():      Global.save\_changes()      Global.rege = 2      Global.logopen = false      self.queue\_free()        pass # Replace with function body.  func \_on\_Movement\_item\_selected(type):       # how to rotate 1 = left and right arrow 2 = a d keys 3=mouse location      Global.userData[14] = str(type)  func \_on\_Fire\_item\_selected(type):      #how to fire 1 = f 2 =space 3 = r 4=mouse click      Global.userData[15] = str(type) |
| Scene |  |

#### Delete v2

|  |  |
| --- | --- |
| What is it? | The only difference is the use of the dat file instead of the txt file. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  func \_ready():      $Label1.hide()  # Called when the node enters the scene tree for the first time.  func \_on\_Settings\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return true      return false  func incorrectLogin():      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $Label1.hide()      $Userbox.text = "Username"      $Passbox.text = "Password"  func \_on\_Delete\_pressed():      var userName = $Userbox.text      var passWord = $Passbox.text      var foundUser = searchUser(userName)      if foundUser == false:          incorrectLogin()          pass      else:          var foundPass = realPass(userName, passWord)          if foundPass == -1:              incorrectLogin()              pass          else:              Global.userData = []              var file = File.new()              file.open("res://userData.dat", file.READ)              var content = file.get\_as\_text()              content = content.split("\n",true)                file.close()              var counter = len(content)-1              for i in len(content):                  var splitContent = content[i].split(",",true)                  if splitContent[0] == userName:                        content[i] = ""                  counter = counter - 1                file.open("res://userData.dat", file.WRITE)              for g in len(content):                  file.store\_line(content[g])              file.close()              Global.cleanup()                  file.open("res://HashPass.dat", file.READ)              content = []              content = file.get\_as\_text().split("\n",true)              print(content)              file.close()              content[foundPass] = "del"              file.open("res://HashPass.dat", file.WRITE)              for i in len(content)-1:                  file.store\_line(str(content[i]))              file.close()              Global.rege = 0              Global.logopen = false              self.queue\_free()      func realPass(user ,passw):      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()        var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              return  num          elif content[num] == "":              return -1          elif content[num] == " ":              return -1          elif content[num] == "end":              num = 0          if count > 10:              return -1          num = num +1  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |
| Scene |  |

#### Game v4

|  |  |
| --- | --- |
| What is it? | Same as last time but also a way to spawn ammo kits and medkits also I put in a new feature that a minimum of 4 zombies are on screen at once. Also decrease the tick speed by making it increase every 3 seconds instead of 2 and will pause if the pause button is pressed. |
| Troubles |  |
| Script | extends Node2D  const HUD = preload("res://Scenes/HUD.tscn")  const Player = preload("res://Scenes/Player.tscn")  const Zombie = preload("res://Scenes/Zombie.tscn")  const AmmoKit = preload("res://Scenes/AmmoKit.tscn")  const MedKit = preload("res://Scenes/MedKit.tscn")  var launchZ = 0  var launchA = 0  var launchM = 0  export (float) var time = 0  #wait between spawns of zombies  func waitz():      yield(get\_tree().create\_timer(1.5), "timeout")      launchZ = 0  #wait between spawns of medkits  func waitm():      yield(get\_tree().create\_timer(20.0), "timeout")      launchM = 0  #wait between spawns of ammokits  func waita():      yield(get\_tree().create\_timer(20.0), "timeout")      launchA = 0  # Called when the node enters the scene tree for the first time.  func \_ready():      Global.pause = 0      Global.game = 1      Global.newGame = 0      Global.ticks = 0      Global.zombieNum = 0      Global.hook = 0      Global.aKit = 0      Global.mKit = 0      var player = Player.instance()      var main2 = get\_tree().current\_scene      main2.add\_child(player)      var HUD = HUD.instance()      var main = get\_tree().current\_scene      main.add\_child(HUD)      var loop = true      while loop == true:          yield(get\_tree().create\_timer(3.0), "timeout")          if Global.pause == 0:              Global.ticks = Global.ticks + 0.5  func \_process(\_delta):      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              Global.logopen = false              self.queue\_free()          pass      else:          if Global.zombieNum < 5:              var zombie = Zombie.instance()              var main = get\_tree().current\_scene              main.add\_child(zombie)              Global.zombieNum =Global.zombieNum  +  1              if launchZ == 0:              if Global.zombieNum < 20:                  var zombie = Zombie.instance()                  var main = get\_tree().current\_scene                  main.add\_child(zombie)                  Global.zombieNum =Global.zombieNum  +  1                  launchZ = 1                  waitz()          if launchM == 0:              if Global.mKit == 0:                  if Global.hp < int(Global.userData[3]):                      var med = MedKit.instance()                      var main = get\_tree().current\_scene                      main.add\_child(med)                      Global.mKit = 1                      launchM = 1                      waitm()          if launchA == 0:              if Global.aKit == 0:                  if Global.ammo < int(Global.userData[2]):                      var ammo = AmmoKit.instance()                      var main = get\_tree().current\_scene                      main.add\_child(ammo)                      Global.aKit = 1                      launchA = 1                      waita() |
| Scene | It is empty as a blank canvas for all the game sprites |

#### HUD v3

|  |  |
| --- | --- |
| What is it? | Same as last time but bump the selector numbers by one and change from txt to dat. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var Hudtype = int(Global.userData[13])  var scores = []  # Called when the node enters the scene tree for the first time.  func \_ready():  $ProHUDHP.set\_max(int(Global.userData[3]))  $ProHUDAmmo.set\_max(int(Global.userData[2]))  Global.hp = int(Global.userData[3])  Global.ammo = int(Global.userData[2])    Global.cash = 0  Global.score = 0  $Unpause.hide()  $Quit.hide()  $Over.hide()  scores.append(int(Global.userData[8]))  scores.append(int(Global.userData[9]))  scores.append(int(Global.userData[10]))  scores.append(int(Global.userData[11]))  scores.append(int(Global.userData[12]))    func \_on\_Pause\_pressed():  $Pause.hide()  Global.pause = 1  $Unpause.show()  $Quit.show()  func \_on\_Unpause\_pressed():  $Pause.show()  Global.pause = 0  $Unpause.hide()  $Quit.hide()  func \_on\_Quit\_pressed():      Global.game = 0      self.queue\_free()  func \_process(\_delta):  $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)  if Hudtype == 1:  $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\nAmmo: "+ str(Global.ammo) + " of "+ Global.userData[2]  $ProHUDHP.hide()  $ProHUDAmmo.hide()  elif Hudtype == 2:  $HUD.hide()  $ProHUDHP.set\_value(Global.hp)  $ProHUDAmmo.set\_value(Global.ammo)  else:  $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]  $ProHUDHP.set\_value(Global.hp)  $ProHUDAmmo.set\_value(Global.ammo)  if Global.hp <= 0:  Global.hp = 0  if Global.pause == 0:  Global.pause = 1  $Pause.hide()  changes()  func \_on\_Menu\_pressed():  Global.game = 0  self.queue\_free()  func \_on\_Restart\_pressed():  Global.newGame = 1  self.queue\_free()  func changes():  $Over.show()  var change = false  scores.append(Global.score)  Global.userData[1] = str(int(Global.userData[1])+Global.cash)  $Over/TotalCash.text = "Total cash is " + str(Global.userData[1])      for i in range(scores.size()-1, -1, -1):  for j in range(1,i+1,1):  var notes = scores[j]  var prevnotes = scores[j-1]  if notes > prevnotes:  var temp = scores[j-1]    scores[j-1] = scores[j]  scores[j] = temp  change = true      for i in len(scores):  $Scores.text = $Scores.text + str(scores[i]) + "\n"    if change == true:  Global.userData[8]= str(scores[0])  Global.userData[9]= str(scores[1])  Global.userData[10]=str(scores[2])  Global.userData[11]=str(scores[3])  Global.userData[12]=str(scores[4])  Global.save\_changes() |
| Scene |  |

#### Shop v2

|  |  |
| --- | --- |
| What is it? | Switch out from using singleton variable to get the cost and use a function to update them all. |
| Troubles |  |
| Script | extends Control  var newHP = int(Global.userData[3])+2  var newHP\_cost = int(newHP\*2)  var newAmmo = int(Global.userData[2])+3  var newAmmo\_cost = int(newAmmo\*1.5)  var newMed = int(Global.userData[4])+1  var newMed\_cost = int(newMed\*2)  var newAmmok = int(Global.userData[5])+7  var newAmmok\_cost = int(newAmmok\*1.5)    func ready():      $NotEnough.hide()      newHP = int(Global.userData[3])+2      newHP\_cost = int(newHP\*3)      newAmmo = int(Global.userData[2])+3      newAmmo\_cost = int(newAmmo\*1.5)      newMed = int(Global.userData[4])+1      newMed\_cost = int(newMed\*2)      newAmmok = int(Global.userData[5])+5      newAmmok\_cost = int(newAmmok\*1.5)    func \_process(\_delta):        $Money.text = "Total Cash:" + str(int(Global.userData[1]))      var hptext = "Currently:" + Global.userData[3] +", upgraded:"+ str(newHP)+ ", upgarde cost:" + str(newHP\_cost)      $Hp.text = str(hptext)      var ammotext = "Upgrade Max Ammo, Currently:" + Global.userData[2] + ", upgraded:"+ str(newAmmo)+ ", upgrade cost:" + str(newAmmo\_cost)      $Ammo.text = str(ammotext)      var med =  "Upgrade Medkit regen ammount, Currently:" + Global.userData[4] +", upgraded:"+str(newMed)+",upgrade cost:" + str(newMed\_cost)      $Med.text = str(med)      var kit = "Upgrade Ammokit regen ammount, Currently:" + Global.userData[5] + ", upgraded:" + str(newAmmok) +",upgrade cost:" + str(newAmmok\_cost)      $Kit.text = str(kit)    #2345  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func change(cost):      Global.userData[1] = str(int(Global.userData[1])-cost)      Global.save\_changes()      pass  func check(cost):        if int(Global.userData[1]) >= cost:          return true      else:          return false  func notEnough():      $NotEnough.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $NotEnough.hide()      pass  func \_on\_AmmoMax\_pressed():      var enoughCash = check(newAmmo\_cost)      if enoughCash == true:          Global.userData[2] = str(int(Global.userData[2]) +5)          change(newAmmo\_cost)          ready()      else:          notEnough()        pass  func \_on\_Ammokit\_pressed():      var enoughCash = check(newAmmok\_cost)      if enoughCash == true:          Global.userData[5] = str(int(Global.userData[5])+3)          change(newAmmok\_cost)            ready()      else:          notEnough()        pass  func \_on\_HP\_pressed():      var enoughCash = check(newHP\_cost)      if enoughCash == true:          Global.userData[3] = str(int(Global.userData[3]) +2)          change(newHP\_cost)          ready()      else:          notEnough()        pass  func \_on\_MedKit\_pressed():      var enoughCash = check(newMed\_cost)        if enoughCash == true:          Global.userData[4] = str(int(Global.userData[4]) +1)          change(newMed\_cost)          ready()      else:          notEnough()        pass    func \_on\_Menu\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free()      pass # Replace with function body. |
| Scene |  |

#### Player v2

|  |  |
| --- | --- |
| What is it? | Same as last version but a new option of using the mouse to rotate and the click to fire and the h key to fire the hook. I also scaled the player larger by a scale of 2 |
| Troubles |  |
| Script | extends KinematicBody2D  const Shell = preload("res://Scenes/Shell.tscn")  const Hook = preload("res://Scenes/Hook.tscn")  export (float) var rotation\_speed = 1  var sensitivity = int(Global.userData[6])  var rotation\_dir = 0  var left = ""  var right = ""  var fire = ""  var mouseLook = false  # Got the correct signals for the player  func \_ready():      if int(Global.userData[14]) == 1:          left = "ui\_right"          right = "ui\_left"      elif int(Global.userData[14]) == 2:          right = "ui\_a"          left = "ui\_d"      else:          mouseLook = true      if int(Global.userData[15]) == 3:          fire = "ui\_r"      elif int(Global.userData[15]) == 2:          fire = "ui\_space"      elif int(Global.userData[15]) == 1 :          fire = "ui\_f"      else:          fire = "click"      pass # Replace with function body.    func \_process(delta):      rotation\_speed = 5 \* sensitivity      rotation\_dir = 0      if Global.game == 0:          self.queue\_free()        if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      elif mouseLook == false:          if Input.is\_action\_pressed(left):              rotation\_dir += 1          if Input.is\_action\_pressed(right):              rotation\_dir -= 1          self.rotation += rotation\_dir \* rotation\_speed \* delta      else:          var target = get\_global\_mouse\_position()            look\_at(target)          self.rotation = self.rotation +90          if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      elif Input.is\_action\_pressed(fire):          if Global.firing == 0:              if Global.ammo > 0:                  Global.firing = 1                  var shell = Shell.instance()                  var main = get\_tree().current\_scene                  main.add\_child(shell)                  shell.global\_transform = $Muzzle.global\_transform                  Global.ammo = Global.ammo - 1      elif Input.is\_action\_pressed("ui\_h"):          if Global.hook == 0:              Global.hook = 1              var hook = Hook.instance()              var main = get\_tree().current\_scene              main.add\_child(hook)              hook.global\_transform = $Muzzle.global\_transform |
| Scene |  |

#### Shell v3

|  |  |
| --- | --- |
| What is it? | Same as last time but made the shell 2 times larger and when it collides the collisionShape2D will scale up from a 2 to a 4 |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = 0  var playing = 0  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.hide()      $Sprite.hide()      yield(get\_tree().create\_timer(0.1), "timeout")      Global.firing = 0        func \_physics\_process(delta):      if Global.game == 0:            self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      velocity += transform.y \* speed      if hit == 0:          collision = move\_and\_collide(velocity \* delta)      if hit == 0:          if collision:              if str(collision.collider.name) == "Zombie":                  hit = 1        if hit == 1:          if playing == 0:                $Bullet.hide()              $Sprite.show()              $CollisionShape2D.scale.x = 4              $CollisionShape2D.scale.y = 4              $Sprite.frame = 0              $Sprite.playing = true              playing = 1    func \_on\_Sprite\_animation\_finished():        Global.firing = 0      self.queue\_free()    func \_on\_Out\_screen\_exited():      Global.firing = 0      self.queue\_free() |
| Scene |  |

#### Hook v1

|  |  |
| --- | --- |
| What is it? | This is on Layer 2 and Mask 2 while the Shell, Tank and zombies are on mask 1 and mask 1., this is done so that the hook and the zombies do not interact with each other. It is used to get the health and ammo kits increasing the Hp and ammo of the player. It is like the shell but only interacts with the medkit and ammo kit. And it has a node that when it is out of screen it self-terminates. |
| Troubles |  |
| Script | extends KinematicBody2D    export (int) var speed = -100  var velocity = Vector2()  var collision  # Called when the node enters the scene tree for the first time.  func \_ready():      self.scale.x = 0.15      self.scale.y = 0.15      pass # Replace with function body.    func \_process(delta):      if Global.game == 0:            self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      velocity += transform.y \* speed        collision = move\_and\_collide(velocity \* delta)        if collision:          if str(collision.collider.name) == "MedKit":              Global.hp = Global.hp + int(Global.userData[4])              if Global.hp > int(Global.userData[3]):                  Global.hp = int(Global.userData[3])              Global.hook = 0              Global.mKit = 0              self.queue\_free()          if str(collision.collider.name) == "AmmoKit":              Global.ammo = Global.ammo + int(Global.userData[5])              if Global.ammo > int(Global.userData[2]):                  Global.ammo = int(Global.userData[2])              Global.hook = 0              Global.aKit = 0              self.queue\_free()      func \_on\_Out\_screen\_exited():      Global.hook = 0      self.queue\_free() |
| Scene |  |

#### Zombie v2

|  |  |
| --- | --- |
| What is it? | It is the same as last time but less stuff that are useless are in the code. And used a function for when it is hit by a shell. |
| Troubles |  |
| Script | extends KinematicBody2D    var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  func \_ready():      rng.randomize()      $Sprite.playing= true      var where = rng.randi\_range(0,1) # 0 means on base 1 = on height      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              var along =  rng.randi\_range(0,600)              position.y = along            else:              position.x = 1024              var along =  rng.randi\_range(0,600)              position.y = along        else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              var along =  rng.randi\_range(0,1024)              position.x = along          else:              position.y = 600              var along =  rng.randi\_range(0,1024)              position.x = along  func \_physics\_process(\_delta):        var target = Vector2(512, 400)      look\_at(target)      var inc = Global.ticks/5      velocity = position.direction\_to(target)\*inc        if Global.game == 0:          self.queue\_free()        if Global.pause == 0:          $Sprite.playing= true      if Global.pause == 1:          $Sprite.playing= false          if Global.newGame == 1:              self.queue\_free()          pass        elif position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)        if collision:          if str(collision.collider.name) == "Shell":              hit()          if str(collision.collider.name) == "Tank":              Global.hp = Global.hp -1              Global.zombieNum = Global.zombieNum-1              self.queue\_free()          func hit():      Global.score = Global.score +1 + int(Global.ticks)      Global.cash = Global.cash +1      Global.zombieNum = Global.zombieNum-1      self.queue\_free() |
| Scene |  |

#### MedKit v1

|  |  |
| --- | --- |
| What is it? | This is a sprite that appears once the hp is below max then will disappear if hit, then will reappear after 10 seconds, if not hit by hook, then will naturally self-terminate after 10 seconds until hp is below max and 10 seconds have passed |
| Troubles | I could not get a way for it to disappear after it was launched as it was stationary so move and collide would not work and would be cumbersome, so I used singleton variables as there would be only one kit of each type on the screen at once so if the variable mKit went to 0 it would cease. |
| Script | extends KinematicBody2D    var rng = RandomNumberGenerator.new()  func \_ready():      rng.randomize()      var whereX = rng.randi\_range(100,924)      var whereY = rng.randi\_range(100,500)        self.position.x = whereX      self.position.y = whereY      yield(get\_tree().create\_timer(10.0), "timeout")      Global.mKit = 0      self.queue\_free()        pass    func \_process(\_delta):          if Global.game == 0:          self.queue\_free()          if Global.pause == 1:            if Global.newGame == 1:              self.queue\_free()          pass        if Global.mKit == 0:          self.queue\_free()        pass |
| Scene |  |

#### AmmoKit v1

|  |  |
| --- | --- |
|  | |
| What is it? | This is a sprite that appears once the ammo is below max then will disappear if hit, then will reappear after 10 seconds, if not hit by hook, then will naturally self-terminate after 10 seconds until ammo is below max and 10 seconds have passed |
| Troubles | I could not get a way for it to disappear after it was launched as it was stationary so move and collide would not work and would be cumbersome, so I used singleton variables as there would be only one kit of each type on the screen at once so if the variable aKit went to 0 it would cease. |
| Script | extends KinematicBody2D  # Declare member variables here. Examples:  # var a = 2  # var b = "text"    var collision  var rng = RandomNumberGenerator.new()  func \_ready():      rng.randomize()      var whereX = rng.randi\_range(100,924)      var whereY = rng.randi\_range(100,500)        self.position.x = whereX      self.position.y = whereY      yield(get\_tree().create\_timer(10.0), "timeout")      Global.aKit = 0      self.queue\_free()        pass # Replace with function body.    func \_process(\_delta):          if Global.game == 0:          self.queue\_free()          if Global.pause == 1:            if Global.newGame == 1:              self.queue\_free()          pass        if Global.aKit == 0:          self.queue\_free()        pass |
| Scene |  |

#### White box testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected | Change | What happens |
| See if the ammo kit and medkit will appear | Ammo and med kit should appear | N/A | It worked. |
| The MedKit worked | The Hp variable would increase when the medkit is hit. |  | Worked   |  | | --- | |  | |  | |
| The AmmoKit worked | The Hp variable would increase when the medkit is hit. |  | Worked   |  | | --- | |  | |  | |
| See if hooked work | If hooks do not interact with the zombies only kits |  | Did it. |
| See if the control scene works | It must display the correct controls |  | Worked |

#### Summary

This iteration overhauls everything from the rest of the prototype; the main scene has a piece of code now to make the hash table if it is not found, if the table was not there the new hash table was not formatted correctly, so this fixes the issue; the score scene was improved using bubble sort fulfilling its success criteria; the settings scene has been updated to allow more rotation and firing keybinds partially fulfilling success criteria as it still doesn’t control volume; the player has had new keybinds added partially fulfilling success criteria as it still doesn’t allow the changing of the users tank.

The new scene control allows the user to see what control their using so do not need to go into settings to edit it, increasing the games usability. The med and ammo kits allow the player to heal and shoot more so fulfils the success criteria of it being a shooter as they are necessary for them. The hook has been added to allow the user to collect the kits.

### Summary of prototype 1

Prototype 1 has finished and fulfils these success criteria.

|  |  |  |
| --- | --- | --- |
| Success Criteria | Fulfilled or not | Where |
| People can make up to 10 accounts on each game instance, so people do not make an infinite amount, filling the storage until there is a computer crash. | Fulfilled | [Registration v1](#_Registration_v1) |
| The player can exit the game from the HUD. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can pause the gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| Make sure login works. | Fulfilled | [Login v2](#_Login_v2) |
| Make sure the password system is secure | Has not been fulfilled, I will probably use lots of if statements and flags to code for it. |  |
| The person can change input settings (key bindings). | Fulfilled | [Settings v2](#_Settings_v2) |
| The person can change volume settings. | Semi-Fulfilled, it’s made but it’s not been used. | [Settings v1](#_Settings_v1) |
| The person can change delete all records of their scores. | Fulfilled | [Delete v1](#_Delete_v1) |
| The person can change the sensitivity of movement. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change reset all progress. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can make it full screen. | Has not been fulfilled, I will probably use a button to toggle it. |  |
| The person can change if they want the HP to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change if they want the ammo to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change and delete their account. | Fulfilled | [Delete v1](#_Delete_v1)  [Settings v1](#_Settings_v1) |
| The player can see the amount of in-game cash they have in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can see the total in-game cash they have on the menu. | Fulfilled | [Menu v1](#_Menu_v1) |
| The player can be able to see their top 5 scores. | Fulfilled | [Scores v2](#_Scores_v2) |
| The player can be able to see the top 5 scores from all accounts. | Fulfilled | [Scores v2](#_Scores_v2) |
| The player can upgrade themselves in the in-game shop with in-game cash. | Fulfilled | [Shop v1](#_Shop_v1) |
| Themed music must play in certain scenes. | Has not been fulfilled, I will have music be played by the HUD and main scene, stopping it if it ever user is one that scene. |  |
| Some lore that is interesting | Has not been fulfilled, I will probably make a video that I will use as a skippable cutscene. |  |
| The enemy moves towards the player and face the player. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The player can rotate | Fulfilled | [Player v1](#_Player_v1) |
| The shell travels in the direction fired | Fulfilled | [Shell v1](#_Shell_v1) |
| The hook travels in the direction fired | Fulfilled | [Hook v1](#_Hook_v1) |
| The kits will appear randomly. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| Upon gameplay, the rotation speed is affected by the sensitivity set in the settings. | Fulfilled | [Player v1](#_Player_v1) |
| The player can fire a tank shell upon the assigned key binding. | Fulfilled | [Shell v1](#_Shell_v1)  [Player v1](#_Player_v1) |
| The player can fire a hook upon assigned key binding. | Fulfilled | [Player v1](#_Player_v1)  [Hook v1](#_Hook_v1) |
| The player’s health must decrease if the enemy touches them and then the enemy disappears. | Fulfilled | [HUD v1](#_HUD_v1) |
| The enemy must disappear if touching shell/explosion and increase score and in-game cash. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The enemies speed and damage done to the player must increase linearly during one gameplay and reset after it. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The enemy must appear randomly around the player. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| Tank-repair-kits and ammo kits must appear randomly on the screen. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| Tank-repair-kits and ammo kits must disappear when touching the hook and increase the appropriate bar. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1)  [Hook v1](#_Hook_v1) |
| Tank-repair-kits and ammo kits must disappear in a set amount of time if it is not used. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| The player is defeated when health is 0. | Fulfilled | [HUD v1](#_HUD_v1)  [Zombie v1](#_Zombie_v1) |
| The player can see enemies. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The player can see the playable character. | Fulfilled | [Player v1](#_Player_v1) |
| The player can interact with the game. | Fulfilled | [Player v1](#_Player_v1) |
| Player can change ammo type | Has not been fulfilled, I will probably make singleton variable and make the pressing of those buttons change it in the HUD. |  |
| Must have at least 2 variants of enemies, normal and boss | Semi-fulfilled, I will probably have them instance from the game scene. |  |
| The player can see their ammo and HP levels. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can see the score they got in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| Must be able to pause and quit the game. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player must be able see the amount of cash they got | Fulfilled | [HUD v3](#_Hud_v3) |

### References for prototype 1

|  |  |
| --- | --- |
| What | Where |
| Making the timer using yield(get\_tree().create\_timer(2.0), "timeout")” | <https://gdscript.com/solutions/godot-timing-tutorial/> |
| Collisions | <https://docs.godotengine.org/en/stable/tutorials/physics/physics_introduction.html> |
| Instancing | Mr. Ngai |
| How to get denary of a character | <https://docs.godotengine.org/en/stable/classes/class_@gdscript.html>? |
| How to use the autoloader/make singleton variables | <https://www.youtube.com/watch?v=5k8xuefBQHs&ab_channel=Davies> |
| Close the game | <https://godotengine.org/qa/554/is-there-a-way-to-close-a-game-using-gdscript> |
| Random numbers | <https://docs.godotengine.org/en/stable/classes/class_randomnumbergenerator.html> |
| How to read and write files | <https://docs.godotengine.org/en/stable/classes/class_file.html?highlight=file> |
| Rotation | <https://docs.godotengine.org/en/stable/tutorials/2d/2d_movement.html> |

## Prototype 2

For this prototype I focused on things that will improve the game experience like music and background images.

### Day 24/05/2021

On this one I made some background images, added a credits section to say all attributions to images and music and tried to remove all unnecessary code.

#### Main v5

|  |  |
| --- | --- |
| What is it? | Same as last time but a credits scene that attributes to any music or picture creator. It now has a background for all other scene to load to and closes when playing the game loop, it has the node background. The image was made by layering different images together in Godot. |
| Troubles | Just a repetition of what I did already. |
| Script | extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  const Credits = preload("res://Scenes/Credits.tscn")  # Declare member variables here. Examples:  # var a = 2  # var b = "text"    # Called when the node enters the scene tree for the first time.  func \_ready():      $Background.show()      get\_tree().paused = false      Global.cleanup()        if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      print(doFileExists)      var remake = []#if the password file doesn't exist it makes one and      if doFileExists == false:#formats it          for i in 34:              print(i)              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()      pass # Replace with function body.        func \_process(\_delta):      if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              $Background.show()              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              $Background.show()              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              $Background.show()              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              $Background.show()              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true      if Global.rege == 4: # rege 4 = scores          if Global.logopen == false:              $Background.show()              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true      if Global.rege == 5: # rege 5 = settings          if Global.logopen == false:              $Background.show()              var setting = Settings.instance()              var main = get\_tree().current\_scene              main.add\_child(setting)              Global.logopen = true      if Global.rege == 6: # rege 6 = account deletion          if Global.logopen == false:              $Background.show()              var delete = Delete.instance()              var main = get\_tree().current\_scene              main.add\_child(delete)              Global.logopen = true      if Global.rege == 7: # rege 7= starts game          if Global.logopen == false:              $Background.hide()              var game = Game.instance()              var main = get\_tree().current\_scene              main.add\_child(game)              Global.logopen = true              get\_tree().paused = false      if Global.rege == 8: # rege 8= starts controls          if Global.logopen == false:              $Background.show()              var controls = Controls.instance()              var main = get\_tree().current\_scene              main.add\_child(controls)              Global.logopen = true      if Global.rege == 9: # rege 8= starts credits          if Global.logopen == false:              $Background.show()              var credits = Credits.instance()              var main = get\_tree().current\_scene              main.add\_child(credits)              Global.logopen = true      pass |
| Scene |  |
| Node Tree |  |

#### Credits v1

|  |  |
| --- | --- |
| What is it? | This scene will allow the user to see the attributions to content creators. As it is only static text it does not need much code. |
| Troubles |  |
| Script | extends Control  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      self.queue\_free() |
| Scene |  |
| Node tree |  |

#### Login v5

|  |  |
| --- | --- |
| What is it? | The only change is in how it looks. |
| Scene |  |
| Node tree |  |

#### Registration v4

|  |  |
| --- | --- |
| What is it? | Same as last time but different look. |
| Scene |  |
| Node tree |  |

#### Menu v4

|  |  |
| --- | --- |
| What is it? | Same as last version but a scene that tells the user the credits. |
| Troubles |  |
| Script | extends Control    func \_ready():      $Label1.hide()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))    func \_on\_Quit\_pressed():      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      get\_tree().quit()  func \_on\_Shop\_pressed():      Global.rege = 3      Global.logopen = false      self.queue\_free()  func \_on\_Game\_pressed():      Global.rege = 7      Global.logopen = false      self.queue\_free()  func \_on\_Settings\_pressed():      Global.rege = 5      Global.logopen = false      self.queue\_free()  func \_on\_Controls\_pressed():      Global.rege = 8      Global.logopen = false      self.queue\_free()  func \_on\_Credits\_pressed():      Global.rege = 9      Global.logopen = false      self.queue\_free()  func \_on\_Scores\_pressed():      Global.rege = 4      Global.logopen = false      self.queue\_free()      pass # Replace with function body. |
| Scene |  |
| Node tree |  |

#### Scores v4

|  |  |
| --- | --- |
| What is it? | Same as last time but looks. |
| Scene |  |
| Node tree |  |

#### Settings v4

|  |  |
| --- | --- |
| What is it? | Same as last time but the volume is now a scrollable slider and removed any unnecessary code like prints, it also looks different. |
| Troubles |  |
| Script | extends Control  func \_ready():      $HUD.add\_item("Change on how you see the HUD in Game")      $HUD.set\_item\_disabled(0, true)      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")      $Movement.add\_item("Change on how you rotate player")      $Movement.set\_item\_disabled(0, true)      $Movement.add\_item("Left and Right arrow keys")      $Movement.add\_item("Using the A and D keys")      $Movement.add\_item("Using the Mouse location")      $Fire.add\_item("Change on how you fire")      $Fire.set\_item\_disabled(0, true)      $Fire.add\_item("The F key")      $Fire.add\_item("The Space bar")      $Fire.add\_item("The R key")      $Fire.add\_item("Using a mouse click")  func \_process(\_delta):      $Sens.value = int(Global.userData[6])      $Vol.value = int(Global.userData[7])      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userData[6])      $Vol/Label.text = "Volume is currently: " + str(Global.userData[7])    func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userData[6] = value    func \_on\_Vol\_value\_changed(value):      value = str(value)      Global.userData[7] = value  func \_on\_Delete\_pressed():      Global.rege = 6      Global.logopen = false      self.queue\_free()  func \_on\_Reset\_pressed():      Global.userData[1] ="0"      Global.userData[2]="20"      Global.userData[3]="10"      Global.userData[4]="2"      Global.userData[5]="10"      Global.userData[6]="1"      Global.userData[7]="50"      Global.userData[8]="0"      Global.userData[9]="0"      Global.userData[10]="0"      Global.userData[11]="0"      Global.userData[12]="0"      Global.userData[13]="3"      Global.userData[14]="1"      Global.userData[15]="1"      Global.save\_changes()      \_on\_Menu\_pressed()  func \_on\_HUD\_item\_selected(type):      Global.userData[13] = str(type)  func \_on\_Menu\_pressed():      Global.save\_changes()      Global.rege = 2      Global.logopen = false      self.queue\_free()      pass  func \_on\_Movement\_item\_selected(type):       # how to rotate 1 = left and right arrow 2 = a d keys 3=mouse location      Global.userData[14] = str(type)  func \_on\_Fire\_item\_selected(type):      #how to fire 1 = f 2 =space 3 = r 4=mouse click      Global.userData[15] = str(type) |
| Scene |  |
| Node tree |  |

#### Delete v3

|  |  |
| --- | --- |
| What is it? | The only difference is the use of the dat file instead of the txt file. |
| Scene |  |
| Node tree |  |

#### Game v5

|  |  |
| --- | --- |
| What is it? | Same code but different look. |
| Scene | A picture containing sunset  Description automatically generated |
| Node tree |  |

#### HUD v4

|  |  |
| --- | --- |
| What is it? | Same but using the native pause feature in Godot and my own one to pause the game. It also updates the account total cash in the changes procedure now. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var Hudtype = int(Global.userData[13])  var scores = []  # Called when the node enters the scene tree for the first time.  func \_ready():      $ProHUDHP.set\_max(int(Global.userData[3]))      $ProHUDAmmo.set\_max(int(Global.userData[2]))      Global.hp = int(Global.userData[3])      Global.ammo = int(Global.userData[2])      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      $Over.hide()        scores.append(int(Global.userData[8]))      scores.append(int(Global.userData[9]))      scores.append(int(Global.userData[10]))      scores.append(int(Global.userData[11]))      scores.append(int(Global.userData[12]))      pass    func \_on\_Pause\_pressed():      $Pause.hide()      Global.pause = 1      get\_tree().paused = true      $Unpause.show()      $Quit.show()  func \_on\_Unpause\_pressed():      $Pause.show()      Global.pause = 0      get\_tree().paused = false      $Unpause.hide()      $Quit.hide()  func \_on\_Quit\_pressed():      Global.game = 0      Global.pause = 0      get\_tree().paused = false      self.queue\_free()  func \_process(\_delta):      $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      if Hudtype == 1:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\nAmmo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.hide()          $ProHUDAmmo.hide()      elif Hudtype == 2:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      else:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      if Global.hp <= 0:          Global.hp = 0          if Global.pause == 0:              get\_tree().paused = true              $Pause.hide()              changes()              Global.pause = 1  func \_on\_Menu\_pressed():      Global.game = 0      get\_tree().paused = false      self.queue\_free()  func \_on\_Restart\_pressed():      Global.newGame = 1      get\_tree().paused = false      self.queue\_free()  func changes():      if Global.pause == 0:          Global.pause = 1          $Over.show()          var change = false          scores.append(Global.score)          Global.userData[1] = str(int(Global.userData[1])+Global.cash)          $Over/TotalCash.text = "Total cash is " + str(Global.userData[1])          for i in range(scores.size()-1, -1, -1):              for j in range(1,i+1,1):                  var notes = scores[j]                  var prevnotes = scores[j-1]                  if notes > prevnotes:                      var temp = scores[j-1]                      scores[j-1] = scores[j]                      scores[j] = temp                      change = true          for i in len(scores):              $Over/Scores.text = $Over/Scores.text + str(scores[i]) + "\n"          if change == true:              Global.userData[8]= str(scores[0])              Global.userData[9]= str(scores[1])              Global.userData[10]=str(scores[2])              Global.userData[11]=str(scores[3])              Global.userData[12]=str(scores[4])          Global.save\_changes() |
| Scene |  |
| Node tree |  |

#### Shop v3

|  |  |
| --- | --- |
| What is it? | Same but different looks |
| Scene |  |
| Node tree |  |

#### Player v3

|  |  |
| --- | --- |
| What is it? | The only change is the size of the player. |
| Scene |  |
| Node tree |  |

#### Zombie v3

|  |  |
| --- | --- |
| What is it? | Same as last time but added a bit that removes any unnecessary details in the collision name stopping any bugs, and if it does it prints the name of the item the colliding object that is weird. I also removed the hit procedure as it was unnecessary. |
| Troubles |  |
| Script | extends KinematicBody2D    var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  func \_ready():      rng.randomize()      $Sprite.playing= true      var where = rng.randi\_range(0,1) # 0 means on base 1 = on height      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              var along =  rng.randi\_range(0,600)              position.y = along            else:              position.x = 1024              var along =  rng.randi\_range(0,600)              position.y = along        else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              var along =  rng.randi\_range(0,1024)              position.x = along          else:              position.y = 600              var along =  rng.randi\_range(0,1024)              position.x = along    func \_physics\_process(\_delta):        var target = Vector2(512, 400)      look\_at(target)      var inc = Global.ticks/5      velocity = position.direction\_to(target)\*inc        if Global.game == 0:          self.queue\_free()        if Global.hp <= 0:          self.queue\_free()      if Global.pause == 1:          $Sprite.playing= false          if Global.newGame == 1:              self.queue\_free()          pass        elif position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)        if Global.pause == 0:          $Sprite.playing= true          if collision:              var item = str(collision.collider.name)              item = item.replace("@","")              item = item.replace("1","")              item = item.replace("2","")              item = item.replace("3","")              item = item.replace("4","")              item = item.replace("5","")              item = item.replace("6","")              item = item.replace("7","")              item = item.replace("8","")              item = item.replace("9","")              item = item.replace("0","")                if item == "Shell":                  Global.score = Global.score +1 + int(Global.ticks)                  Global.cash = Global.cash +1                  Global.zombieNum = Global.zombieNum-1                  self.queue\_free()              elif item == "Tank":                  Global.hp = Global.hp -1                  Global.zombieNum = Global.zombieNum-1                  self.queue\_free()              elif item != "Zombie":                  print(item) |
| Scene |  |
| Node tree |  |

#### White box testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Change from previous | Expected | Outcome | Evidence |
| See if the background nodes in the main scene hides when the game launches |  | It hides when game launches | It hid. |  |
| The background nodes reappear when game ends. |  | The nodes reappear. | It showed. |  |
| See if the current code works |  | Everything should go as previously. | Everything was going OK, but the ammo wasn’t draining when the player fired. |  |
| Retest for if the ammo system works properly. | The code was missing the line “Global.ammo = Global.ammo -1” It was there in previous versions, may have removed it by accident. | The ammo would drain if the player fires | It worked |  |
| Is the credits scene show when their button is pressed? |  | The menu would disappear, and credits would show. | It appeared. |  |

#### Summary

This iteration deals with aesthetics and efficiency: I added a background for when logged in and when not logged in. I also removed all most print functions and streamlined the collision detection in the zombie. The credits scene attributes any places I got media for the game.

### Day 25/06/2021

On this day I added music and sound effects.

#### Main v6

|  |  |
| --- | --- |
| What is it? | Same as last time but a background music will play when the player logs in, the volume can be edited in the settings. The music is stopped when the player starts the game. For the volume I use a formula to get the correct volume for the game. |
| Troubles | I didn’t know how to use the audio node but then I looked at the documents and I realized it was quite simple of telling it to play or not. |
| Script | extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  const Credits = preload("res://Scenes/Credits.tscn")  var playing = 1  # Declare member variables here. Examples:  # var a = 2  # var b = "text"    # Called when the node enters the scene tree for the first time.  func \_ready():      OS.set\_window\_title("Zombie Shooter By Arman")      $Background.show()      get\_tree().paused = false      Global.cleanup()        if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      print(doFileExists)      var remake = []      if doFileExists == false:          for i in 34:              print(i)              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()      pass # Replace with function body.  func \_on\_Music\_finished():      if Global.game == 0:          $Music.play()      else:          $Music.stop()      func \_process(\_delta):      if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              $Background.show()              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true              if int(Global.userData[7]) == 0:                  $Music.stop()              else:                  $Music.play()              if int(Global.userData[7]) == 50:                  $Music.volume\_db = -25              elif int(Global.userData[7]) < 50:                  var num = 1.0                  num = int(Global.userData[7]) \*0.5                  num = -50 + num                  $Music.volume\_db = num              elif int(Global.userData[7]) > 50:                  var num = 1.0                  num = (int(Global.userData[7])-50) \*0.5                  num = -25 + num                  $Music.volume\_db = num      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              $Background.show()              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              $Background.show()              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              $Background.show()              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true      if Global.rege == 4: # rege 4 = scores          if Global.logopen == false:              $Background.show()              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true      if Global.rege == 5: # rege 5 = settings          if Global.logopen == false:              $Background.show()              var setting = Settings.instance()              var main = get\_tree().current\_scene              main.add\_child(setting)              Global.logopen = true      if Global.rege == 6: # rege 6 = account deletion          if Global.logopen == false:              $Background.show()              var delete = Delete.instance()              var main = get\_tree().current\_scene              main.add\_child(delete)              Global.logopen = true      if Global.rege == 7: # rege 7= starts game          if Global.logopen == false:              $Background.hide()              var game = Game.instance()              var main = get\_tree().current\_scene              main.add\_child(game)              Global.logopen = true              get\_tree().paused = false              $Music.stop()      if Global.rege == 8: # rege 8= starts controls          if Global.logopen == false:              $Background.show()              var controls = Controls.instance()              var main = get\_tree().current\_scene              main.add\_child(controls)              Global.logopen = true        if Global.rege == 9: # rege 8= starts credits          if Global.logopen == false:              $Background.show()              var credits = Credits.instance()              var main = get\_tree().current\_scene              main.add\_child(credits)              Global.logopen = true        pass |
| Scene |  |
| Node Tree |  |

#### Credits v2

|  |  |
| --- | --- |
| What is it? | Same as last time but more people are attributed because of the music and sound effects I am using. |
| Scene |  |

#### AmmoKit v2

|  |  |
| --- | --- |
| What is it? | Same as last time but a sound effect if it touches the hook. but I did not code for it yet. Only put in the node. |
| Node tree |  |

#### MedKit v2

|  |  |
| --- | --- |
| What is it? | Same as last time but a sound effect if it touches the hook. but I did not code for it yet. Only put in the node. |
| Node tree |  |

#### Hook v2

|  |  |
| --- | --- |
| What is it? | Same as last time but with a better sprite. |
| Scene |  |
| Node tree |  |

#### HUD v5

|  |  |
| --- | --- |
| What is it? | Same as last time but a sound effect when the player dies. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var Hudtype = int(Global.userData[13])  var scores = []  # Called when the node enters the scene tree for the first time.  func \_ready():      $ProHUDHP.set\_max(int(Global.userData[3]))      $ProHUDAmmo.set\_max(int(Global.userData[2]))      Global.hp = int(Global.userData[3])      Global.ammo = int(Global.userData[2])      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      $Over.hide()        scores.append(int(Global.userData[8]))      scores.append(int(Global.userData[9]))      scores.append(int(Global.userData[10]))      scores.append(int(Global.userData[11]))      scores.append(int(Global.userData[12]))      pass    func \_on\_Pause\_pressed():      $Pause.hide()      Global.pause = 1      get\_tree().paused = true      $Unpause.show()      $Quit.show()  func \_on\_Unpause\_pressed():      $Pause.show()      Global.pause = 0      get\_tree().paused = false      $Unpause.hide()      $Quit.hide()  func \_on\_Quit\_pressed():      Global.game = 0      Global.pause = 0      get\_tree().paused = false      self.queue\_free()  func \_process(\_delta):      $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      if Hudtype == 1:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\nAmmo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.hide()          $ProHUDAmmo.hide()      elif Hudtype == 2:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      else:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      if Global.hp <= 0:          Global.hp = 0          if Global.pause == 0:              get\_tree().paused = true              $Pause.hide()              changes()              Global.pause = 1              $Death.play()  func \_on\_Menu\_pressed():      Global.game = 0      get\_tree().paused = false      self.queue\_free()  func \_on\_Restart\_pressed():      Global.newGame = 1      get\_tree().paused = false      self.queue\_free()  func changes():      if Global.pause == 0:          Global.pause = 1          $Over.show()          var change = false          scores.append(Global.score)          Global.userData[1] = str(int(Global.userData[1])+Global.cash)          $Over/TotalCash.text = "Total cash is " + str(Global.userData[1])          for i in range(scores.size()-1, -1, -1):              for j in range(1,i+1,1):                  var notes = scores[j]                  var prevnotes = scores[j-1]                  if notes > prevnotes:                      var temp = scores[j-1]                      scores[j-1] = scores[j]                      scores[j] = temp                      change = true          for i in len(scores):              $Over/Scores.text = $Over/Scores.text + str(scores[i]) + "\n"          if change == true:              Global.userData[8]= str(scores[0])              Global.userData[9]= str(scores[1])              Global.userData[10]=str(scores[2])              Global.userData[11]=str(scores[3])              Global.userData[12]=str(scores[4])          Global.save\_changes() |
| Scene |  |
| Node tree |  |

#### Player v4

|  |  |
| --- | --- |
| What is it? | A sound effect when the player fires a shot. It also modulates the sound to the volume requested. |
| Troubles |  |
| Script | extends KinematicBody2D  const Shell = preload("res://Scenes/Shell.tscn")  const Hook = preload("res://Scenes/Hook.tscn")  export (float) var rotation\_speed = 1  var sensitivity = int(Global.userData[6])  var rotation\_dir = 0  var left = ""  var right = ""  var fire = ""  var mouseLook = false  # Got the correct signals for the player  func \_ready():      if int(Global.userData[14]) == 1:          left = "ui\_right"          right = "ui\_left"      elif int(Global.userData[14]) == 2:          right = "ui\_a"          left = "ui\_d"      else:          mouseLook = true      if int(Global.userData[15]) == 3:          fire = "ui\_r"      elif int(Global.userData[15]) == 2:          fire = "ui\_space"      elif int(Global.userData[15]) == 1 :          fire = "ui\_f"      else:          fire = "click"      if int(Global.userData[7]) == 50:          $SFXFire.volume\_db = -25      elif int(Global.userData[7]) < 50:          var num = 1.0          num = int(Global.userData[7]) \*0.5          num = -50 + num          $SFXFire.volume\_db = num      elif int(Global.userData[7]) > 50:          var num = 1.0          num = (int(Global.userData[7])-50) \*0.5          num = -25 + num          $SFXFire.volume\_db = num      pass # Replace with function body.    func \_process(delta):      rotation\_speed = 5 \* sensitivity      rotation\_dir = 0      if Global.game == 0:          self.queue\_free()        if Global.newGame == 1:          self.queue\_free()      elif mouseLook == false:          if Input.is\_action\_pressed(left):              rotation\_dir += 1          if Input.is\_action\_pressed(right):              rotation\_dir -= 1          self.rotation += rotation\_dir \* rotation\_speed \* delta        else:          var target = get\_global\_mouse\_position()          look\_at(target)          self.rotation = self.rotation +90            if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      if Input.is\_action\_pressed(fire):          if Global.firing == 0:              if Global.ammo > 0:                  Global.firing = 1                  var shell = Shell.instance()                  var main = get\_tree().current\_scene                  main.add\_child(shell)                  shell.global\_transform = $Muzzle.global\_transform                  Global.ammo = Global.ammo -1                  if int(Global.userData[7]) == 0:                      $SFXFire.stop()                  else:                      $SFXFire.play()          if Input.is\_action\_pressed("ui\_h"):          if Global.hook == 0:              Global.hook = 1              var hook = Hook.instance()              var main = get\_tree().current\_scene              main.add\_child(hook)              hook.global\_transform = $Muzzle.global\_transform |
| Scene |  |
| Node tree |  |

#### Zombie v4

|  |  |
| --- | --- |
| What is it? | A sound effect of when it dies or when it touches the tank. |
| Troubles |  |
| Script | extends KinematicBody2D    var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  var hit = 0  func \_ready():      rng.randomize()      $Sprite.playing= true      var where = rng.randi\_range(0,1) # 0 means on base 1 = on height      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              var along =  rng.randi\_range(0,600)              position.y = along            else:              position.x = 1024              var along =  rng.randi\_range(0,600)              position.y = along        else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              var along =  rng.randi\_range(0,1024)              position.x = along          else:              position.y = 600              var along =  rng.randi\_range(0,1024)              position.x = along      if int(Global.userData[7]) == 50:          $SFXEat.volume\_db = -25          $SFXDie.volume\_db = -25      elif int(Global.userData[7]) < 50:          var num = 1.0          num = int(Global.userData[7]) \*0.5          num = -50 + num          $SFXEat.volume\_db = num          $SFXDie.volume\_db = num      elif int(Global.userData[7]) > 50:          var num = 1.0          num = (int(Global.userData[7])-50) \*0.5          num = -25 + num          $SFXEat.volume\_db = num          $SFXDie.volume\_db = num    func \_physics\_process(\_delta):        var target = Vector2(512, 400)      look\_at(target)      var inc = Global.ticks/5      velocity = position.direction\_to(target)\*inc        if Global.game == 0:          self.queue\_free()        if Global.hp <= 0:          self.queue\_free()      if Global.pause == 1:          $Sprite.playing= false          if Global.newGame == 1:              self.queue\_free()          pass        elif position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)        if Global.pause == 0:          $Sprite.playing= true          if collision:              var item = str(collision.collider.name)              item = item.replace("@","")              item = item.replace("1","")              item = item.replace("2","")              item = item.replace("3","")              item = item.replace("4","")              item = item.replace("5","")              item = item.replace("6","")              item = item.replace("7","")              item = item.replace("8","")              item = item.replace("9","")              item = item.replace("0","")              if hit ==0:                  if item == "Shell":                      hit = 1                      Global.score = Global.score +1 + int(Global.ticks)                      Global.cash = Global.cash +1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXDie.play()                      yield(get\_tree().create\_timer(0.45), "timeout")                        self.queue\_free()                  elif item == "Tank":                      hit = 1                      Global.hp = Global.hp -1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXEat.play()                      yield(get\_tree().create\_timer(1.3), "timeout")                        self.queue\_free()                  elif item != "Zombie":                      print(item) |
| Scene |  |
| Node tree |  |

#### White box testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Change from previous | Expected | Outcome | Evidence |
| If the music played when logged in. |  | Music will play. | It played |  |
| If the music will stop playing when the game starts |  | Music will stop. | It stopped. |  |
| When the shell is fired there will be a SFX. |  | The SFX will play when the shell is fired. | It played. |  |
| The zombie will make the SFX of eating when it touches the tank. |  | Play the SFXEat node. | Played |  |
| The zombie will make the SFX when touching tank. |  | Play the SFXDie | It played. |  |

#### Summary

This iteration deals with mainly music and sound effects(sfx), it partially fulfils the criteria of having themed music as the menu music is themed to be calm and relaxing, it also fulfils the criteria for volume control. The sfx played helps user usability as there is now sound verification to visual verification now. The settings volume changes has now been tested so now successful.

### Day 26/06/2021

Added more SFX.

#### Main v7

|  |  |
| --- | --- |
| What is it? | Added a button clicked SFX. |
| Troubles |  |
| Script | extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  const Credits = preload("res://Scenes/Credits.tscn")  var playing = 0  # Declare member variables here. Examples:  # var a = 2  # var b = "text"    # Called when the node enters the scene tree for the first time.  func \_ready():        OS.set\_window\_title("Zombie Shooter By Arman")      $Background.show()      get\_tree().paused = false      Global.cleanup()        if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      print(doFileExists)      var remake = []      if doFileExists == false:          for i in 34:              print(i)              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()        pass # Replace with function body.  func \_on\_Music\_finished():      if Global.game == 0:          $Music.play()      else:          $Music.stop()    func play():      $SFXButton.play()  func \_process(\_delta):      if Global.button == true:          Global.button = false          play()      if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              $Background.show()              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true              if int(Global.userData[7]) == 0:                  $Music.stop()              else:                  if playing == 0:                      $Music.play()                      playing = 1              if int(Global.userData[7]) == 50:                  $Music.volume\_db = -25              elif int(Global.userData[7]) < 50:                  var num = 1.0                  num = int(Global.userData[7]) \*0.5                  num = -50 + num                  $Music.volume\_db = num              elif int(Global.userData[7]) > 50:                  var num = 1.0                  num = (int(Global.userData[7])-50) \*0.5                  num = -25 + num                  $Music.volume\_db = num      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              $Background.show()              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              $Background.show()              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              $Background.show()              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true        if Global.rege == 4: # rege 4 = scores          if Global.logopen == false:              $Background.show()              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true        if Global.rege == 5: # rege 5 = settings          if Global.logopen == false:              $Background.show()              var setting = Settings.instance()              var main = get\_tree().current\_scene              main.add\_child(setting)              Global.logopen = true        if Global.rege == 6: # rege 6 = account deletion          if Global.logopen == false:              $Background.show()              var delete = Delete.instance()              var main = get\_tree().current\_scene              main.add\_child(delete)              Global.logopen = true        if Global.rege == 7: # rege 7= starts game          if Global.logopen == false:              $Background.hide()              var game = Game.instance()              var main = get\_tree().current\_scene              main.add\_child(game)              Global.logopen = true              get\_tree().paused = false              $Music.stop()              playing = 0      if Global.rege == 8: # rege 8= starts controls          if Global.logopen == false:              $Background.show()              var controls = Controls.instance()              var main = get\_tree().current\_scene              main.add\_child(controls)              Global.logopen = true        if Global.rege == 9: # rege 8= starts credits          if Global.logopen == false:              $Background.show()              var credits = Credits.instance()              var main = get\_tree().current\_scene              main.add\_child(credits)              Global.logopen = true        pass |
| Scene |  |
| Node Tree |  |

#### AmmoKit v3

|  |  |
| --- | --- |
| What is it? | I hook up the code to the audio node. The volume is controlled by the player. |
| Troubles |  |
| Script | extends KinematicBody2D  var collision  var hit =0  var rng = RandomNumberGenerator.new()  func \_ready():      if int(Global.userData[7]) == 50:          $Get.volume\_db = -25      elif int(Global.userData[7]) < 50:          var num = 1.0          num = int(Global.userData[7]) \*0.5          num = -50 + num          $Get.volume\_db = num      elif int(Global.userData[7]) > 50:          var num = 1.0          num = (int(Global.userData[7])-50) \*0.5          num = -25 + num          $Get.volume\_db = num      rng.randomize()      var whereX = rng.randi\_range(100,924)      var whereY = rng.randi\_range(100,500)        self.position.x = whereX      self.position.y = whereY      yield(get\_tree().create\_timer(10.0), "timeout")      Global.aKit = 0      self.queue\_free()      pass    func \_process(\_delta):      if Global.game == 0:          self.queue\_free()      if Global.newGame == 1:          self.queue\_free()      if hit == 0:          if Global.aKit == 0:              hit = 1              self.hide()              $Get.play()              yield(get\_tree().create\_timer(0.5), "timeout")              self.queue\_free()      pass |
| Scene |  |
| Node tree |  |

#### MedKit v3

|  |  |
| --- | --- |
| What is it? | I hook up the code to the audio node. The volume is controlled by the player. |
| Troubles |  |
| Script | extends KinematicBody2D    var hit =0  var rng = RandomNumberGenerator.new()  func \_ready():      if int(Global.userData[7]) == 50:          $Get.volume\_db = -25      elif int(Global.userData[7]) < 50:          var num = 1.0          num = int(Global.userData[7]) \*0.5          num = -50 + num          $Get.volume\_db = num      elif int(Global.userData[7]) > 50:          var num = 1.0          num = (int(Global.userData[7])-50) \*0.5          num = -25 + num          $Get.volume\_db = num      rng.randomize()      var whereX = rng.randi\_range(100,924)      var whereY = rng.randi\_range(100,500)        self.position.x = whereX      self.position.y = whereY      yield(get\_tree().create\_timer(10.0), "timeout")      Global.mKit = 0      self.queue\_free()      pass  func \_process(\_delta):      if Global.game == 0:          self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      if hit == 0:          if Global.mKit == 0:              hit = 1              self.hide()              $Get.play()              yield(get\_tree().create\_timer(0.76), "timeout")              self.queue\_free()        pass |
| Scene |  |
| Node tree |  |

#### Zombie v5

|  |  |
| --- | --- |
| What is it? | Added effect of the zombies modulating their colour randomly |
| Troubles |  |
| Script | extends KinematicBody2D    var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  var hit = 0  func \_ready():      rng.randomize()      $Sprite.playing= true      var where = rng.randi\_range(0,1) # 0 means on base 1 = on height      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              var along =  rng.randi\_range(0,600)              position.y = along            else:              position.x = 1024              var along =  rng.randi\_range(0,600)              position.y = along        else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              var along =  rng.randi\_range(0,1024)              position.x = along          else:              position.y = 600              var along =  rng.randi\_range(0,1024)              position.x = along      if int(Global.userData[7]) == 50:          $SFXEat.volume\_db = -25          $SFXDie.volume\_db = -25      elif int(Global.userData[7]) < 50:          var num = 1.0          num = int(Global.userData[7]) \*0.5          num = -50 + num          $SFXEat.volume\_db = num          $SFXDie.volume\_db = num      elif int(Global.userData[7]) > 50:          var num = 1.0          num = (int(Global.userData[7])-50) \*0.5          num = -25 + num          $SFXEat.volume\_db = num          $SFXDie.volume\_db = num      var r = rng.randf\_range(0.5,1)      var g = rng.randf\_range(0.5,1)      var b = rng.randf\_range(0.5,1)      $Sprite.modulate = Color(r,g,b)  func \_physics\_process(\_delta):        var target = Vector2(512, 400)      look\_at(target)      var inc = Global.ticks/5      velocity = position.direction\_to(target)\*inc        if Global.game == 0:          self.queue\_free()        if Global.hp <= 0:          self.queue\_free()      if Global.pause == 1:          $Sprite.playing= false          if Global.newGame == 1:              self.queue\_free()          pass        elif position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)        if Global.pause == 0:          $Sprite.playing= true          if collision:              var item = str(collision.collider.name)              item = item.replace("@","")              item = item.replace("1","")              item = item.replace("2","")              item = item.replace("3","")              item = item.replace("4","")              item = item.replace("5","")              item = item.replace("6","")              item = item.replace("7","")              item = item.replace("8","")              item = item.replace("9","")              item = item.replace("0","")              if hit ==0:                  if item == "Shell":                      hit = 1                      Global.score = Global.score +1 + int(Global.ticks)                      Global.cash = Global.cash +1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXDie.play()                      yield(get\_tree().create\_timer(0.45), "timeout")                        self.queue\_free()                  elif item == "Tank":                      hit = 1                      Global.hp = Global.hp -1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXEat.play()                      yield(get\_tree().create\_timer(1.3), "timeout")                        self.queue\_free()                  elif item != "Zombie":                      print(item) |
| Scene |  |
| Node tree | Graphical user interface, text, application  Description automatically generated |

#### White box testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Change from previous | Expected | Outcome | Evidence |
| If a button is pressed the button sound effect occurs. |  | SFX will play | It played |  |
| If the medkit is hooked it plays SFX. |  | When hooked the medkit plays its SFX. | Played |  |
| If the ammo kit is hooked it plays SFX. |  | When hooked the ammo kit plays its SFX | Played. |  |

#### Summary

This iteration just added the sfx to the kits and the buttons. It also slightly changes the zombie’s colour by modulating it, making each enemy unique slightly.

### Summary for prototype 2

I polished up the game and made it more enjoyable in this prototype by adding music.

|  |  |  |
| --- | --- | --- |
| Success Criteria | Fulfilled or not | Where |
| People can make up to 10 accounts on each game instance, so people do not make an infinite amount, filling the storage until there is a computer crash. | Fulfilled | [Registration v1](#_Registration_v1) |
| The player can exit the game from the HUD. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can pause the gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| Make sure login works. | Fulfilled | [Login v2](#_Login_v2) |
| Make sure the password system is secure | Has not been fulfilled, I will probably use lots of if statements and flags to code for it. |  |
| The person can change input settings (key bindings). | Fulfilled | [Settings v2](#_Settings_v2) |
| The person can change volume settings. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change delete all records of their scores. | Fulfilled | [Delete v1](#_Delete_v1) |
| The person can change the sensitivity of movement. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change reset all progress. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can make it full screen. | Has not been fulfilled, I will probably use a button to toggle it. |  |
| The person can change if they want the HP to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change if they want the ammo to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change and delete their account. | Fulfilled | [Delete v1](#_Delete_v1)  [Settings v1](#_Settings_v1) |
| The player can see the amount of in-game cash they have in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can see the total in-game cash they have on the menu. | Fulfilled | [Menu v1](#_Menu_v1) |
| The player can be able to see their top 5 scores. | Fulfilled | [Scores v2](#_Scores_v2) |
| The player can be able to see the top 5 scores from all accounts. | Fulfilled | [Scores v2](#_Scores_v2) |
| The player can upgrade themselves in the in-game shop with in-game cash. | Fulfilled | [Shop v1](#_Shop_v1) |
| Themed music must play in certain scenes. | Partially fulfilled, will probably have music be played by the HUD as game music. | [Main v6](#_Main_v6) |
| Some lore that is interesting | Has not been fulfilled, I will probably make a video that I will use as a skippable cutscene. |  |
| The enemy moves towards the player and face the player. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The player can rotate | Fulfilled | [Player v1](#_Player_v1) |
| The shell travels in the direction fired | Fulfilled | [Shell v1](#_Shell_v1) |
| The hook travels in the direction fired | Fulfilled | [Hook v1](#_Hook_v1) |
| The kits will appear randomly. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| Upon gameplay, the rotation speed is affected by the sensitivity set in the settings. | Fulfilled | [Player v1](#_Player_v1) |
| The player can fire a tank shell upon the assigned key binding. | Fulfilled | [Shell v1](#_Shell_v1)  [Player v1](#_Player_v1) |
| The player can fire a hook upon assigned key binding. | Fulfilled | [Player v1](#_Player_v1)  [Hook v1](#_Hook_v1) |
| The player’s health must decrease if the enemy touches them and then the enemy disappears. | Fulfilled | [HUD v1](#_HUD_v1) |
| The enemy must disappear if touching shell/explosion and increase score and in-game cash. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The enemies speed and damage done to the player must increase linearly during one gameplay and reset after it. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The enemy must appear randomly around the player. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| Tank-repair-kits and ammo kits must appear randomly on the screen. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| Tank-repair-kits and ammo kits must disappear when touching the hook and increase the appropriate bar. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1)  [Hook v1](#_Hook_v1) |
| Tank-repair-kits and ammo kits must disappear in a set amount of time if it is not used. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| The player is defeated when health is 0. | Fulfilled | [HUD v1](#_HUD_v1)  [Zombie v1](#_Zombie_v1) |
| The player can see enemies. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The player can see the playable character. | Fulfilled | [Player v1](#_Player_v1) |
| The player can interact with the game. | Fulfilled | [Player v1](#_Player_v1) |
| Player can change ammo type | Has not been fulfilled, I will probably make singleton variable and make the pressing of those buttons change it in the HUD. |  |
| Must have at least 2 variants of enemies, normal and boss | Semi-fulfilled, I will probably have them instance from the game scene. |  |
| The player can see their ammo and HP levels. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can see the score they got in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| Must be able to pause and quit the game. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player must be able see the amount of cash they got | Fulfilled | [HUD v3](#_Hud_v3) |

### References for prototype 2

|  |  |
| --- | --- |
| What | Where |
| Audio control | <https://docs.godotengine.org/en/stable/tutorials/audio/audio_streams.html> |

## Prototype 3

Putting in the finishing touches for the game and making everything better.

### Day 14/07/2021

Added some finishing touches and stakeholder requests.

#### Main v8

|  |  |
| --- | --- |
| What is it? | Just added one more instance for the lore scene. I also added a bit that changes the windows title to "Zombie Shooter By Arman". |
| Troubles | Had to look up how to set a title, and how to set volume. |
| Script | extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  const Credits = preload("res://Scenes/Credits.tscn")  const Lore = preload("res://Scenes/Lore.tscn")  var playing = 0  # Declare member variables here. Examples:  # var a = 2  # var b = "text"    # Called when the node enters the scene tree for the first time.  func \_ready():        OS.set\_window\_title("Zombie Shooter By Arman")      $Background.show()      get\_tree().paused = false      Global.cleanup()      if Global.logged == false:          var login = Login.instance()          var main = get\_tree().current\_scene          main.add\_child(login)          Global.logopen = true      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      print(doFileExists)      var remake = []      if doFileExists == false:          for i in 34:              print(i)              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()      pass  func \_on\_Music\_finished():      if Global.game == 0:          $Music.play()      else:          $Music.stop()  func play():      $SFXButton.play()  func \_process(\_delta):      if Global.button == true:          Global.button = false          play()        if Global.rege == 2: # rege 2 = menu          if Global.logopen == false:              $Background.show()              var menu = Menu.instance()              var main = get\_tree().current\_scene              main.add\_child(menu)              Global.logopen = true              if int(Global.userData[7]) == 0:                  $Music.stop()              else:                  if playing == 0:                      $Music.play()                      playing = 1              if int(Global.userData[7]) == 50:                  $Music.volume\_db = -25              elif int(Global.userData[7]) < 50:                  var num = 1.0                  num = int(Global.userData[7]) \*0.5                  num = -50 + num                  $Music.volume\_db = num              elif int(Global.userData[7]) > 50:                  var num = 1.0                  num = (int(Global.userData[7])-50) \*0.5                  num = -25 + num                  $Music.volume\_db = num      if Global.rege == 1: # rege 1 = register          if Global.logopen == false:              $Background.show()              var reges = Registration.instance()              var main = get\_tree().current\_scene              main.add\_child(reges)              Global.logopen = true      if Global.rege == 0: # rege 0 = Login          if Global.logopen == false:              $Background.show()              var login = Login.instance()              var main = get\_tree().current\_scene              main.add\_child(login)              Global.logopen = true      if Global.rege == 3: # rege 3 = shop          if Global.logopen == false:              $Background.show()              var shop = Shop.instance()              var main = get\_tree().current\_scene              main.add\_child(shop)              Global.logopen = true      if Global.rege == 4: # rege 4 = scores          if Global.logopen == false:              $Background.show()              var scores = Scores.instance()              var main = get\_tree().current\_scene              main.add\_child(scores)              Global.logopen = true      if Global.rege == 5: # rege 5 = settings          if Global.logopen == false:              $Background.show()              var setting = Settings.instance()              var main = get\_tree().current\_scene              main.add\_child(setting)              Global.logopen = true      if Global.rege == 6: # rege 6 = account deletion          if Global.logopen == false:              $Background.show()              var delete = Delete.instance()              var main = get\_tree().current\_scene              main.add\_child(delete)              Global.logopen = true      if Global.rege == 7: # rege 7= starts game          if Global.logopen == false:              $Background.hide()              var game = Game.instance()              var main = get\_tree().current\_scene              main.add\_child(game)              Global.logopen = true              get\_tree().paused = false              $Music.stop()              playing = 0      if Global.rege == 8: # rege 8= starts controls          if Global.logopen == false:              $Background.show()              var controls = Controls.instance()              var main = get\_tree().current\_scene              main.add\_child(controls)              Global.logopen = true      if Global.rege == 9: # rege 9= starts credits          if Global.logopen == false:              $Background.show()              var credits = Credits.instance()              var main = get\_tree().current\_scene              main.add\_child(credits)              Global.logopen = true      if Global.rege == 10: # rege 10= starts lore          if Global.logopen == false:              $Background.show()              var lore = Lore.instance()              var main = get\_tree().current\_scene              main.add\_child(lore)              Global.logopen = true      pass |
| Scene | A picture containing indoor, remote, dark  Description automatically generated |
| Node Tree |  |

#### Login v6

|  |  |
| --- | --- |
| What is it? | The only change is the section that codes for the usage of the button sound effect. |
| Troubles |  |
| Script | extends Control  func \_ready():      $Label2.hide()      $Label1.hide()      pass  func \_on\_Log\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")  #checks to see if login is correct  #if yes it move on if not then  #it will show the Label2  #which tells us that the login is incorrect      var user = $Userbox.text      var find = searchUser(user)      if find == false:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 1          Global.logopen = false          self.queue\_free()      var passw = $Passbox.text      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()        var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              Global.logged = true              break          elif content[num] == "":              break          elif content[num] == " ":              break          elif content[num] == "end":              num = 0          if count > 10:              return false          num = num +1      if Global.logged == true:          $Label1.show()          yield(get\_tree().create\_timer(0.5), "timeout")          Global.rege = 2          Global.logopen = false          self.queue\_free()      else:          $Label2.show()          yield(get\_tree().create\_timer(2.0), "timeout")          Global.rege = 0          Global.logopen = false          self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              Global.userData = userline              return true          if  userline[0] == "":              return false          if  userline[0] == " ":              return false      return true  func \_on\_Reg\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 1      Global.logopen = false      self.queue\_free()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |

#### Registration v5

|  |  |  |
| --- | --- | --- |
| What is it? | Change in some settings, like volume is now 0 to 100, so set to 50 now and rotate type. | |
| Troubles |  | |
| Script | | |
| extends Control    #files in userData saved in format of username,cash,shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10),normal\_sensitivity(1),sound(1-100=volume, 0 =false)  #,highscore1,highscore2,highscore3,highscore4,highscore5,type of HUD 1 = only text 2 = progress bar 3 = both, how to rotate 1 = left and right arrow 2 = a d keys 3 =mouse position,  #how to fire 1 = f 2 =space 2 = r 3= click\n  func \_ready():      $Label1.hide()      $Label2.hide()      Global.cleanup()      var file = File.new()      var count = 0      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var arrcontent = content.split("\n")      for n in len(arrcontent):          count += 1        if count >= 11:          $Label1.show()          $Reg.hide()          $Userbox.hide()          $Passbox.hide()          $ConPassbox.hide()      pass # Replace with function body.    func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return false          if  userline[0] == "":              return true          if  userline[0] == " ":              return true      return true  func passCheck(passw):      if passw == "del":          return false      if passw == " ":          return false      if passw == "":          return false      if passw == "end":          return false      return true    func addLogin(user, passw):      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var file2 = File.new()      file2.open("res://userData.dat", file.WRITE)      var add = content+"\n"+user +",0,20,10,2,10,1,50,0,0,0,0,0,3,1,1"      file2.store\_line(add)      file2.close()      var loco = Global.hashedLocation(user)      var hashed = passw.sha256\_text()      var file3 = File.new()      file3.open("res://HashPass.dat", file.READ)      var content2 = file3.get\_as\_text()      content2 = content2.split("\n", true)        file3.close()      var file4 = File.new()      file4.open("res://HashPass.dat", file.WRITE)      content2[loco] = hashed      for i in len(content2)-1:          file4.store\_line(str(content2[i]))      file4.close()  func wrong():      $Label2.visible = visible      yield(get\_tree().create\_timer(2.0), "timeout")      $Label2.visible = !visible  func \_on\_Reg\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      var user = $Userbox.text      var passw = $Passbox.text      var findu = searchUser(user)      if $Passbox.text == $ConPassbox.text:          if findu == false:              wrong()              pass          if findu == true:              var findp = passCheck(passw)              if findp == false:                  wrong()                  pass              else:                  addLogin(user,passw)                  Global.cleanup()                  Global.rege = 0                  Global.logopen = false                  self.queue\_free()      else:          wrong()          pass  func \_on\_Back\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 0      Global.logopen = false      self.queue\_free() | |

#### Lore v1

|  |  |
| --- | --- |
| What is it? | Just some pictures and some text, explaining the history of the person playing. |
| Script | extends Control  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      self.queue\_free() |
| Scene |  |
| Node tree |  |

#### Controls v2

|  |  |
| --- | --- |
| What is it? | Added how to switch between shell types and the button noise. |
| Troubles |  |
| Script | extends Control  var movement = ""  var shootS = ""  var hook = "When the H key is pressed"  func \_ready():      if int(Global.userData[14]) == 1:          movement = "When the Left and Right Arrow Keys are pressed"      elif int(Global.userData[14]) == 2:          movement = "When the A and D keys are pressed"      else:          movement = "Using the global mouse position"      if int(Global.userData[15]) == 3:          shootS = "When the R key is pressed"      elif int(Global.userData[15]) == 2:          shootS = "When the Space bar is pressed"      elif int(Global.userData[15]) == 1 :          shootS = "When the F key is pressed"      else:          shootS = "When the mouse is clicked"      var text = " Movement of player is:"+ movement+ "\n How to shoot the Shell: "+ shootS + "\n How fire the hook: " + hook      text = text + "\n press 1 in game for normal round, 2 for incendiary round and 3 for Large normal round"      text = text + "\n normal takes 1 ammo, incendiary takes 2 ammo and large takes 3 ammo"      $Stuff.text = text      pass  func \_on\_Menu\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 2      Global.logopen = false      self.queue\_free()      pass |
| Scene |  |

#### Credits v3

|  |  |
| --- | --- |
| What is it? | Same as last time but more people are attributed because of the music and sound effects I am using. |
| Scene |  |

#### Delete v4

|  |  |
| --- | --- |
| What is it? | Only the sound effect button activator. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  func \_ready():      $Label1.hide()  func \_on\_Settings\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 2      Global.logopen = false      self.queue\_free()  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return true      return false  func incorrectLogin():      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $Label1.hide()      $Userbox.text = "Username"      $Passbox.text = "Password"  func \_on\_Delete\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      var userName = $Userbox.text      var passWord = $Passbox.text      var foundUser = searchUser(userName)      if foundUser == false:          incorrectLogin()          pass      else:          var foundPass = realPass(userName, passWord)          if foundPass == -1:              incorrectLogin()              pass          else:              Global.userData = []              var file = File.new()              file.open("res://userData.dat", file.READ)              var content = file.get\_as\_text()              content = content.split("\n",true)                file.close()              var counter = len(content)-1              for i in len(content):                  var splitContent = content[i].split(",",true)                  if splitContent[0] == userName:                        content[i] = ""                  counter = counter - 1                file.open("res://userData.dat", file.WRITE)              for g in len(content):                  file.store\_line(content[g])              file.close()              Global.cleanup()                  file.open("res://HashPass.dat", file.READ)              content = []              content = file.get\_as\_text().split("\n",true)              print(content)              file.close()              content[foundPass] = "del"              file.open("res://HashPass.dat", file.WRITE)              for i in len(content)-1:                  file.store\_line(str(content[i]))              file.close()              Global.rege = 0              Global.logopen = false              self.queue\_free()      func realPass(user ,passw):      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      print(content)      file.close()        var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          print(count)          if content[num] == hashed:              return  num          elif content[num] == "":              return -1          elif content[num] == " ":              return -1          elif content[num] == "end":              num = 0          if count > 10:              return -1          num = num +1  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass |

#### HUD v6

|  |  |
| --- | --- |
| What is it? | Same as last time but a section now tells what type of ammo the player is using currently. And also, the sound effect code. |
| Troubles |  |
| Script | extends Control  var Hudtype = int(Global.userData[13])  var scores = []  var getCash = Global.cash  # Called when the node enters the scene tree for the first time.  func \_ready():      $ProHUDHP.set\_max(int(Global.userData[3]))      $ProHUDAmmo.set\_max(int(Global.userData[2]))      Global.hp = int(Global.userData[3])      Global.ammo = int(Global.userData[2])      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      $Over.hide()        scores.append(int(Global.userData[8]))      scores.append(int(Global.userData[9]))      scores.append(int(Global.userData[10]))      scores.append(int(Global.userData[11]))      scores.append(int(Global.userData[12]))      pass    func \_on\_Pause\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      $Pause.hide()      Global.pause = 1      get\_tree().paused = true      $Unpause.show()      $Quit.show()  func \_on\_Unpause\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      $Pause.show()      Global.pause = 0      get\_tree().paused = false      $Unpause.hide()      $Quit.hide()  func \_on\_Quit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.game = 0      Global.pause = 0      get\_tree().paused = false      self.queue\_free()  func \_process(\_delta):      $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      if Global.typeShot == 1:          $TypePic.frame = 0          $TypeText.text = "Normal"      if Global.typeShot == 2:          $TypePic.frame = 1          $TypeText.text = "Incendiary"      if Global.typeShot == 3:          $TypePic.frame = 2          $TypeText.text = "Large"        if Hudtype == 1:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\nAmmo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.hide()          $ProHUDAmmo.hide()      elif Hudtype == 2:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      else:          $HUD.text = " HP: " + str(Global.hp) + " of "+Global.userData[3] +"\n Ammo: "+ str(Global.ammo) + " of "+ Global.userData[2]          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      if Global.hp <= 0:          Global.hp = 0          if Global.pause == 0:              get\_tree().paused = true                $Pause.hide()                changes()              Global.pause = 1                $Death.play()    func \_on\_Menu\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.game = 0      get\_tree().paused = false      self.queue\_free()  func \_on\_Restart\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.newGame = 1      get\_tree().paused = false      self.queue\_free()  func changes():        if Global.pause == 0:          Global.pause = 1          $Over.show()          var change = false          scores.append(Global.score)            Global.userData[1] = str(int(Global.userData[1])+Global.cash)          Global.cash = 0            $Over/TotalCash.text = "Total cash is " + str(Global.userData[1])          for i in range(scores.size()-1, -1, -1):              for j in range(1,i+1,1):                  var notes = scores[j]                  var prevnotes = scores[j-1]                  if notes > prevnotes:                      var temp = scores[j-1]                      scores[j-1] = scores[j]                      scores[j] = temp                      change = true          for i in len(scores):              $Over/Scores.text = $Over/Scores.text + str(scores[i]) + "\n"          if change == true:              Global.userData[8]= str(scores[0])              Global.userData[9]= str(scores[1])              Global.userData[10]=str(scores[2])              Global.userData[11]=str(scores[3])              Global.userData[12]=str(scores[4])          Global.save\_changes()    func \_input(event):      if event.is\_action\_pressed("ui\_1"):          Global.typeShot = 1      if event.is\_action\_pressed("ui\_2"):          Global.typeShot = 2      if event.is\_action\_pressed("ui\_3"):          Global.typeShot = 3 |
| Scene | |  | | --- | |  | |  | |  | |
| Node tree | |  | | --- | |  | |  | |

#### Menu v5

|  |  |
| --- | --- |
| What is it? | Same as last version but a scene that tells the user the lore. |
| Troubles |  |
| Script | extends Control  func \_ready():      $Label1.hide()      $Money.text = "Total Cash:" + str(int(Global.userData[1]))    func \_on\_Quit\_pressed():      Global.button = true      $Label1.show()      yield(get\_tree().create\_timer(2.0), "timeout")      get\_tree().quit()  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func \_on\_Shop\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 3      Global.logopen = false      self.queue\_free()  func \_on\_Game\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 7      Global.logopen = false      self.queue\_free()  func \_on\_Settings\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 5      Global.logopen = false      self.queue\_free()  func \_on\_Controls\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 8      Global.logopen = false      self.queue\_free()  func \_on\_Credits\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 9      Global.logopen = false      self.queue\_free()  func \_on\_Scores\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 4      Global.logopen = false      self.queue\_free()      pass # Replace with function body.  func \_on\_Lore\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 10      Global.logopen = false      self.queue\_free()      pass # Replace with function body. |
| Scene |  |
| Node tree |  |

#### Player v5

|  |  |
| --- | --- |
| What is it? | I added a scope, and I got the tank to play an explosion animation when player has lost, this is done on a separate script. I also added an option to change to different shell types. I also fixed my mouse control option and shortened it. I also moved the code that is controlling the player to its own input function. Shell frequency is also now controlled here. |
| Troubles |  |
| Script “Player” | extends KinematicBody2D  const Shell = preload("res://Scenes/Shell.tscn")  const Hook = preload("res://Scenes/Hook.tscn")  const ShellFire = preload("res://Scenes/ShellFire.tscn")  const ShellLarge = preload("res://Scenes/ShellLarge.tscn")  export (float) var rotation\_speed = 1  var sensitivity = int(Global.userData[6])  var rotation\_dir = 0  var left = ""  var right = ""  var fire = ""  var mouseLook = false  # Got the correct signals for the player  func \_ready():      if int(Global.userData[14]) == 1:          left = "ui\_right"          right = "ui\_left"      elif int(Global.userData[14]) == 2:          right = "ui\_a"          left = "ui\_d"      else:          mouseLook = true      if int(Global.userData[15]) == 3:          fire = "ui\_r"      elif int(Global.userData[15]) == 2:          fire = "ui\_space"      elif int(Global.userData[15]) == 1 :          fire = "ui\_f"      else:          fire = "click"      if int(Global.userData[7]) == 50:          $SFXFire.volume\_db = -25      elif int(Global.userData[7]) < 50:          var num = 1.0          num = int(Global.userData[7]) \*0.5          num = -50 + num          $SFXFire.volume\_db = num      elif int(Global.userData[7]) > 50:          var num = 1.0          num = (int(Global.userData[7])-50) \*0.5          num = -25 + num          $SFXFire.volume\_db = num      pass # Replace with function body.    func \_process(delta):      rotation\_speed = 5 \* sensitivity      rotation\_dir = 0      if Global.game == 0:          self.queue\_free()      if Global.newGame == 1:          self.queue\_free()      elif mouseLook == false:          if Input.is\_action\_pressed(left):              rotation\_dir += 1          if Input.is\_action\_pressed(right):              rotation\_dir -= 1          self.rotation += rotation\_dir \* rotation\_speed \* delta      else:          look\_at(get\_global\_mouse\_position())      func \_input(event):      if event.is\_action\_pressed(fire):          if Global.pause == 0:              if Global.firing == 0:                  print(Global.typeShot)                  if Global.typeShot == 1:                      if Global.ammo > 0:                          Global.firing = 1                          var shell = Shell.instance()                          var main = get\_tree().current\_scene                          main.add\_child(shell)                          shell.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -1                          if int(Global.userData[7]) == 0:                              $SFXFire.stop()                          else:                              $SFXFire.play()                          yield(get\_tree().create\_timer(0.3), "timeout")                          Global.firing = 0                  elif Global.typeShot == 2:                      if Global.ammo > 1:                          Global.firing = 1                          var shellfire = ShellFire.instance()                          var main = get\_tree().current\_scene                          main.add\_child(shellfire)                          shellfire.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -2                          if int(Global.userData[7]) == 0:                              $SFXFire.stop()                          else:                              $SFXFire.play()                          yield(get\_tree().create\_timer(0.6), "timeout")                          Global.firing = 0                  elif Global.typeShot == 3:                      if Global.ammo > 2:                          Global.firing = 1                          var shelllarge = ShellLarge.instance()                          var main = get\_tree().current\_scene                          main.add\_child(shelllarge)                          shelllarge.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -3                          if int(Global.userData[7]) == 0:                              $SFXFire.stop()                          else:                              $SFXFire.play()                          yield(get\_tree().create\_timer(0.9), "timeout")                          Global.firing = 0      if event.is\_action\_pressed("ui\_h"):          if Global.pause == 0:              if Global.hook == 0:                  Global.hook = 1                  var hook = Hook.instance()                  var main = get\_tree().current\_scene                  main.add\_child(hook)                  hook.global\_transform = $Muzzle.global\_transform |
| Script “Tank\_dark” | extends AnimatedSprite  func \_ready():      self.animation = "default"      self.playing = false      self.show()      $Label.show()      pass  func \_process(\_delta):      if Global.hp <= 0:          if Global.pause == 0:              $Label.hide()              self.animation = "blow"              self.playing = true      pass  func \_on\_Tank\_dark\_animation\_finished():      self.hide()      pass |
| Scene |  |
| Node tree |  |

#### Scores v5

|  |  |
| --- | --- |
| What is it? | Added sound effect button. |
| Troubles |  |
| Script | extends Control  # Declare member variables here. Examples:  # var a = 2  # var b = "text"  var allScores =[]  # Called when the node enters the scene tree for the first time.  func \_ready():      $PersonalScores.text = "1. "+Global.userData[8]+"\n"+"2. "+Global.userData[9]+"\n"+"3. "+Global.userData[10]+"\n"+"4. "+Global.userData[11]+"\n"+"5. "+Global.userData[12]+"\n"      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      print(content)      for i in len(content)-1:          var line = content[i]          line = line.split(",", true)          print(line)          var add1 = line[0]+". "+line[8]          var add2 = line[0]+". "+line[9]          var add3 = line[0]+". "+line[10]          var add4 = line[0]+". "+line[11]          var add5 = line[0]+". "+line[12]          allScores.append(add1)          allScores.append(add2)          allScores.append(add3)          allScores.append(add4)          allScores.append(add5)        for i in range(allScores.size()-1, -1, -1):          for j in range(1,i+1,1):              var notes = allScores[j].split(". ", true)              var prevnotes = allScores[j-1].split(". ", true)              if int(notes[1]) > int(prevnotes[1]):                  var temp = allScores[j-1]                  allScores[j-1] = allScores[j]                  allScores[j] = temp      print(allScores)      var scoresOrdered = ""      for i in len(allScores):          scoresOrdered = scoresOrdered + allScores[i] + "\n"      $HighScores.text = scoresOrdered  func \_on\_Back\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 2      Global.logopen = false      self.queue\_free() |

#### Settings v5

|  |  |
| --- | --- |
| What is it? | Same as last time but the added a way to toggle Fullscreen and button sound effect. Some data on the resetting function also change as I start using them. |
| Troubles |  |
| Script | extends Control  func \_ready():      $HUD.add\_item("Change on how you see the HUD in Game")      $HUD.set\_item\_disabled(0, true)      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")      $Movement.add\_item("Change on how you rotate player")      $Movement.set\_item\_disabled(0, true)      $Movement.add\_item("Left and Right arrow keys")      $Movement.add\_item("Using the A and D keys")      $Movement.add\_item("Using the Mouse location")      $Fire.add\_item("Change on how you fire")      $Fire.set\_item\_disabled(0, true)      $Fire.add\_item("The F key")      $Fire.add\_item("The Space bar")      $Fire.add\_item("The R key")      $Fire.add\_item("Using a mouse click")  func \_process(\_delta):      $Sens.value = int(Global.userData[6])      $Vol.value = int(Global.userData[7])      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userData[6])      $Vol/Label.text = "Volume is currently: " + str(Global.userData[7])    func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userData[6] = value    func \_on\_Vol\_value\_changed(value):      value = str(value)      Global.userData[7] = value  func \_on\_Delete\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 6      Global.logopen = false      self.queue\_free()  func \_on\_Reset\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.userData[1] ="0"      Global.userData[2]="20"      Global.userData[3]="10"      Global.userData[4]="2"      Global.userData[5]="10"      Global.userData[6]="1"      Global.userData[7]="50"      Global.userData[8]="0"      Global.userData[9]="0"      Global.userData[10]="0"      Global.userData[11]="0"      Global.userData[12]="0"      Global.userData[13]="3"      Global.userData[14]="1"      Global.userData[15]="1"      Global.save\_changes()      \_on\_Menu\_pressed()  func \_on\_HUD\_item\_selected(type):      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.userData[13] = str(type)  func \_on\_Menu\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.save\_changes()      Global.rege = 2      Global.logopen = false      self.queue\_free()        pass # Replace with function body.  func \_on\_Movement\_item\_selected(type):      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")       # how to rotate 1 = left and right arrow 2 = a d keys 3=mouse location      Global.userData[14] = str(type)  func \_on\_Fire\_item\_selected(type):      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      #how to fire 1 = f 2 =space 3 = r 4=mouse click      Global.userData[15] = str(type)  func \_on\_Fullscreen\_pressed():      OS.window\_fullscreen = !OS.window\_fullscreen      pass # Replace with function body. |
| Scene |  |
| Node tree |  |

#### Shell v4

|  |  |
| --- | --- |
| What is it? | Same as last time but I started using the player script to control intervals between shells. |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = 0  var playing = 0  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.hide()      $Sprite.hide()          func \_physics\_process(delta):      if Global.game == 0:            self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      velocity += transform.y \* speed      if hit == 0:          collision = move\_and\_collide(velocity \* delta)      if hit == 0:          if collision:              if str(collision.collider.name) == "Zombie":                  hit = 1        if hit == 1:          if playing == 0:                $Bullet.hide()              $Sprite.show()              $CollisionShape2D.scale.x = 4              $CollisionShape2D.scale.y = 4              $Sprite.frame = 0              $Sprite.playing = true              playing = 1    func \_on\_Sprite\_animation\_finished():      self.queue\_free()    func \_on\_Out\_screen\_exited():      self.queue\_free() |
| Scene | A picture containing chart  Description automatically generated |
| Node tree |  |

#### ShellFire v1

|  |  |
| --- | --- |
| What is it? | Similar to the Shell scene but now it stays longer on screen for a AoE, by making using a different animation that takes longer. |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = 0  var playing = 0  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.hide()      $Sprite.hide()          func \_physics\_process(delta):      if Global.game == 0:            self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      velocity += transform.y \* speed      if hit == 0:          collision = move\_and\_collide(velocity \* delta)      if hit == 0:          if collision:              if str(collision.collider.name) == "Zombie":                  hit = 1        if hit == 1:          if playing == 0:                $Bullet.hide()              $Sprite.show()              $CollisionShape2D.scale.x = 4              $CollisionShape2D.scale.y = 4              $Sprite.frame = 0              $Sprite.playing = true              playing = 1    func \_on\_Sprite\_animation\_finished():        self.queue\_free()    func \_on\_Out\_screen\_exited():      self.queue\_free() |
| Scene | A picture containing chart  Description automatically generated |
| Node tree | Graphical user interface, application  Description automatically generated |

#### ShellLarge v1

|  |  |
| --- | --- |
| What is it? | Similar to the Shell scene but the AoE is 2 times as big. |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = 0  var playing = 0  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.hide()      $Sprite.hide()          func \_physics\_process(delta):      if Global.game == 0:          self.queue\_free()      if Global.pause == 1:          if Global.newGame == 1:              self.queue\_free()          pass      velocity += transform.y \* speed      if hit == 0:          collision = move\_and\_collide(velocity \* delta)      if hit == 0:          if collision:              if str(collision.collider.name) == "Zombie":                  hit = 1      if hit == 1:          if playing == 0:              $Bullet.hide()              $Sprite.show()              $CollisionShape2D.scale.x = 8              $CollisionShape2D.scale.y = 8              $Sprite.frame = 0              $Sprite.playing = true              playing = 1    func \_on\_Sprite\_animation\_finished():        self.queue\_free()    func \_on\_Out\_screen\_exited():      self.queue\_free() |
| Scene | A picture containing chart  Description automatically generated |
| Node tree | Graphical user interface, application  Description automatically generated |

#### Shop v4

|  |  |
| --- | --- |
| What is it? | Same as last time but now it has button sound effects. |
| Troubles |  |
| Script | extends Control  var newHP = int(Global.userData[3])+2  var newHP\_cost = int(newHP\*2)  var newAmmo = int(Global.userData[2])+3  var newAmmo\_cost = int(newAmmo\*1.5)  var newMed = int(Global.userData[4])+1  var newMed\_cost = int(newMed\*2)  var newAmmok = int(Global.userData[5])+7  var newAmmok\_cost = int(newAmmok\*1.5)    func ready():      $NotEnough.hide()      newHP = int(Global.userData[3])+2      newHP\_cost = int(newHP\*3)      newAmmo = int(Global.userData[2])+3      newAmmo\_cost = int(newAmmo\*1.5)      newMed = int(Global.userData[4])+1      newMed\_cost = int(newMed\*2)      newAmmok = int(Global.userData[5])+5      newAmmok\_cost = int(newAmmok\*1.5)    func \_process(\_delta):        $Money.text = "Total Cash:" + str(int(Global.userData[1]))      var hptext = "Currently:" + Global.userData[3] +", upgraded:"+ str(newHP)+ ", upgarde cost:" + str(newHP\_cost)      $Hp.text = str(hptext)      var ammotext = "Upgrade Max Ammo, Currently:" + Global.userData[2] + ", upgraded:"+ str(newAmmo)+ ", upgrade cost:" + str(newAmmo\_cost)      $Ammo.text = str(ammotext)      var med =  "Upgrade Medkit regen ammount, Currently:" + Global.userData[4] +", upgraded:"+str(newMed)+",upgrade cost:" + str(newMed\_cost)      $Med.text = str(med)      var kit = "Upgrade Ammokit regen ammount, Currently:" + Global.userData[5] + ", upgraded:" + str(newAmmok) +",upgrade cost:" + str(newAmmok\_cost)      $Kit.text = str(kit)    #2345  #shellMax(20),HPMax(10),hpKitsize(2),ammokitSize(10)  func change(cost):      Global.userData[1] = str(int(Global.userData[1])-cost)      Global.save\_changes()      pass  func check(cost):        if int(Global.userData[1]) >= cost:          return true      else:          return false  func notEnough():      $NotEnough.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $NotEnough.hide()      pass  func \_on\_AmmoMax\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      var enoughCash = check(newAmmo\_cost)      if enoughCash == true:          Global.userData[2] = str(int(Global.userData[2]) +5)          change(newAmmo\_cost)          ready()      else:          notEnough()      pass  func \_on\_Ammokit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      var enoughCash = check(newAmmok\_cost)      if enoughCash == true:          Global.userData[5] = str(int(Global.userData[5])+3)          change(newAmmok\_cost)          ready()      else:          notEnough()      pass  func \_on\_HP\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      var enoughCash = check(newHP\_cost)      if enoughCash == true:          Global.userData[3] = str(int(Global.userData[3]) +2)          change(newHP\_cost)          ready()      else:          notEnough()      pass  func \_on\_MedKit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      var enoughCash = check(newMed\_cost)      if enoughCash == true:          Global.userData[4] = str(int(Global.userData[4]) +1)          change(newMed\_cost)          ready()      else:          notEnough()      pass    func \_on\_Menu\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.14), "timeout")      Global.rege = 2      Global.logopen = false      self.queue\_free()      pass # Replace with function body. |

#### globals v7

|  |  |
| --- | --- |
| What is it? | Same as last time but more variables. |
| Troubles | Same as last version |
| Script | |
| extends Node  var logopen = false  var rege = 0  var logged = false  var userData = []  var shopOpen = false  var pause = 0 # if 1 means pause game play, if 0 means let gameplay continue  var game = 0 # 0 means no game, 1 means currently playing  var hp = 1  var ammo = 1  var cash = 0  var score = 0  var newGame = 0  var firing = 0  var zombieNum = 0  export (float) var ticks = 0  var hook = 0  var mKit = 0  var aKit = 0  var button = false  var typeShot = 1 # 1-normal 2-incendiary 3 large-Normal    func cleanup():      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1        file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()              func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)        file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32        var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1            if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1    func save\_changes():      Global.userData[1] = str(int(Global.userData[1])+Global.cash)      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = Global.userData[0]+","+Global.userData[1]+","+Global.userData[2]+","+Global.userData[3]+","+Global.userData[4]+","+Global.userData[5]+","+Global.userData[6]+","+Global.userData[7]+","+Global.userData[8]+","+Global.userData[9]+","+Global.userData[10]+","+Global.userData[11]+","+Global.userData[12]+","+Global.userData[13]+","+Global.userData[14]+","+Global.userData[15]      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == Global.userData[0]:                content[i] = replacing          counter = counter - 1        file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      cleanup()  # Called every frame. 'delta' is the elapsed time since the previous frame.  #func \_process(delta):  #   pass | |

#### White box testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Change from previous | Expected | Outcome | Evidence |
| If a button is pressed the button sound effect occurs. |  | SFX will play | It played |  |
| The title change to "Zombie Shooter By Arman" |  | Title changes to "Zombie Shooter By Arman". | Did it |  |
| Does tank play explosion animation |  | The animation plays. | It did. |  |
| If pressing the 2 button the shell type will change to ShellFire |  | Shell type will change to ShellFire | Did it |  |
| If pressing the 3 button the shell type will change to ShellLarge |  | Shell type will change to ShellLarge | Did it |  |
| If pressing the 1 button the shell type will change to Shell |  | Shell type will change to Shel | Did it |  |
| The ShellFire stays as a long term AoE attack |  | Stays in position for the designated time. | It does. |  |
| The ShellLarge gives a large attack |  | Have a large AoE attack by ShellLarge | It does |  |

#### Summary

This iteration just adds a story(lore) scene and the different ammo types, successfully fulfilling the criteria of having different ammo types. The lore however is only partially done so only semi-fulfils its purpose. The settings can now be used to toggle full screen fulfilling one of the success criteria.

### Day 24/07/2021

Tried to make it more user-friendly and used feedback from end user to make improvements in nearly everything. I also increased the resolution of the game 1024x600. I also added a boss mob that spawns every 60 seconds with a hp bar that is equal to the current score. Also added a way to change hook key bind

#### Main v9

|  |  |
| --- | --- |
| What is it? | Changed how the background image is run, it has 6 images and selects randomly one of them to use as the background base, then an image of a boss zombie is facing down in the centre top. There are also the 2 tanks on both sides of the bottom that point towards the boss zombie, there is 9 different tanks that it picks at random. The shell that is pointed to the tank is also random from the image of Shell, ShellFire and ShellLarge. The backdrop changes whenever the app is run, when the menu is opened, and when the game starts. I also use a new way of audio control, so it takes less code, it just controls the main audio bus instead of the individual audio nodes. I also allowed the usage of the m key to toggle Fullscreen when logged in. The zombie and tanks will hide when the game is playing. I also shortened the instancing part of the script |
| Troubles | Looked and found the new audio mechanic. |
| Script | extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  const Credits = preload("res://Scenes/Credits.tscn")  const Lore = preload("res://Scenes/Lore.tscn")  var rng = RandomNumberGenerator.new()  func \_ready():      Global.rege = 0      Backdrop()      OS.set\_window\_title("Zombie Shooter By Arman")      $Background.show()      get\_tree().paused = false      Global.cleanup()      if Global.logged == false:          self.add\_child(Login.instance())          Global.logopen = true      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      var remake = []      if doFileExists == false:          for i in 34:              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()    func backdrop():      rng.randomize()      $Background.frame = rng.randi\_range(0,5)      $Objects/RightTank.frame = rng.randi\_range(0,8)      $Objects/LeftTank.frame = rng.randi\_range(0,8)      $Objects/RightTank/Bullet.frame = rng.randi\_range(0,5)      $Objects/LeftTank/Bullet.frame = rng.randi\_range(0,5)    func \_on\_Music\_finished():      if Global.game == 0:          $Music.play()      else:          $Music.stop()    func \_process(\_delta):      if Global.logged == true:          AudioServer.set\_bus\_volume\_db(AudioServer.get\_bus\_index("Master"), Global.userInfo.getVolume())        if Global.rege != 7:          $Objects.show()          Input.set\_mouse\_mode(Input.MOUSE\_MODE\_VISIBLE)      var flag = false      if Global.logged == true:          if str(Global.userInfo.getVolume()) != "-1.#INF":              if Global.rege != 0:                  if Global.rege != 1:                      if Global.rege != 7:                          if Global.rege != 8:                              if Global.rege != 10:                                  $Music.stream\_paused = false                                  flag = true                                  if $Music.playing == false:                                      $Music.play()      if flag == false:          $Music.stream\_paused = true      if Global.logopen == false:          Global.logopen = true          backdrop()          if Global.rege == 0:              self.add\_child(Login.instance())              $Music.stop()              if OS.window\_fullscreen == true:                  OS.window\_fullscreen = !OS.window\_fullscreen          elif Global.rege == 1:              self.add\_child(Registration.instance())              $Music.stop()          elif Global.rege == 2:              self.add\_child(Menu.instance())          elif Global.rege == 3:              self.add\_child(Shop.instance())          elif Global.rege == 4:              self.add\_child(Scores.instance())          elif Global.rege == 5:              self.add\_child(Settings.instance())          elif Global.rege == 6:              self.add\_child(Delete.instance())          elif Global.rege == 7:              self.add\_child(Game.instance())              backdrop()              $Objects.hide()              get\_tree().paused = false          elif Global.rege == 8:              self.add\_child(Controls.instance())          elif Global.rege == 9:              self.add\_child(Credits.instance())          elif Global.rege == 10:              self.add\_child(Lore.instance())      if Global.button == true:          Global.button = false          if Global.rege != 0:              if Global.rege != 1:                  $SFXButton.play()      pass  func \_input(event):      if Global.logged == true:          if Global.rege !=6:              if event.is\_action\_pressed("ui\_m"):                  OS.window\_fullscreen = !OS.window\_fullscreen |
| Scene |  |
| Node Tree |  |

#### globals v8

|  |  |
| --- | --- |
| What is it? | Same as last time instead of using a list to store a user’s data I use a class and also more variables, also added a way to get info about the data by calling the class and also added the hook change keybind section. |
| Troubles | Had to relearn how to use a class. |
| Script | |
| extends Node  var logopen = false  var rege = 0  var logged = false  var pause = 0  var hp = 1  var ammo = 1  var cash = 0  var score = 0  var newGame = 0  var firing = 0  var zombieNum = 0  export (float) var ticks = 0  var hook = 0  var mKit = 0  var aKit = 0  var button = false  var typeShot = 1  var userInfo  var boss = false  func cleanup():      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1      file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()  class user:      var name      var cash      var maxShell      var maxHP      var kitHP      var kitAmmo      var sens      var volume      var h1      var h2      var h3      var h4      var h5      var hudType      var rotateType      var fireType      var colour      var hook      func \_init(data):          name = data[0]          cash = data[1]          maxShell = data[2]          maxHP = data[3]          kitHP = data[4]          kitAmmo = data[5]          sens = data[6]          volume = data[7]          h1 = data[8]          h2 = data[9]          h3 = data[10]          h4 = data[11]          h5 = data[12]          Hudtype = data[13]          rotateType =data[14]          fireType = data[15]          colour = data[16]          hook = data[17]      func getName():          return name      func getCash():          return int(cash)      func setCash(new):          cash = new      func getMaxShell():          return int(maxShell)      func setMaxShell(new):          maxShell = new      func getMaxHP():          return int(maxHP)      func setMaxHP(new):          maxHP = new      func getkitHP():          return int(kitHP)      func setkitHP(new):          kitHP = new      func getkitAmmo():          return int(kitAmmo)      func setkitAmmo(new):          kitAmmo = new      func getSens():          return int(sens)      func setSens(new):          sens = new      func getVolume():          return int(volume)      func setVolume(new):          volume = new      func getH1():          return h1      func setH1(new):          h1 = new      func getH2():          return h2      func setH2(new):          h2 = new      func getH3():          return h3      func setH3(new):          h3 = new      func getH4():          return h4      func setH4(new):          h4 = new      func getH5():          return h5      func setH5(new):          h5 = new      func getHud():          return int(hudType)      func setHud(new):          hudType = new      func getRotate():          return int(rotateType)      func setRotate(new):          rotateType = new      func getFire():          return int(fireType)      func setFire(new):          fireType = new      func getColour():          return int(colour)      func setColour(new):          colour = new      func getHook():          return int(hook)      func setHook(new):          hook = new      func getALL():          var string =  str(name)+","+str(cash)+","+str(maxShell)+","+str(maxHP)+","+str(kitHP)+","+str(kitAmmo)+","+str(sens)+","+str(volume)+","+str(h1)+","+str(h2)+","+str(h3)+","+str(h4)+","+str(h5)+","+str(Hudtype)+","+str(rotateType)+","+str(fireType)+","+str(colour)+","+str(hook)          return string  func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1  func save\_changes():      Global.userInfo.setCash(Global.userInfo.getCash()+Global.cash)      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = userInfo.getALL()      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == userInfo.getName():              content[i] = replacing          counter = counter - 1      file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      cleanup() | |

#### Login v7

|  |  |
| --- | --- |
| What is it? | I changed from using TextEdit to LineEdit in here as you can make placeholder text, make the password box’s text into a secret and you can use tab to move between them. I also put in a new button for the user to close the application and allowed the use of the enter key to login. |
| Troubles |  |
| Script | extends Control  func \_ready():      $Incorrect.hide()      $LoggedIn.hide()      $Userbox.editable = true      $Passbox.editable = true      $Log.disabled = false      $Quit.disabled = false      $Reg.disabled = false      $Passbox.text = ""      $Userbox.text = ""      pass  func \_on\_Log\_pressed():      Global.button = true      $Userbox.editable = false      $Passbox.editable = false      $Log.disabled = true      $Quit.disabled = true      $Reg.disabled = true      var user = $Userbox.text      var find = searchUser(user)      if find == false:          $Incorrect.show()          yield(get\_tree().create\_timer(2.0), "timeout")          \_ready()          return      var passw = $Passbox.text      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      file.close()      var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          if content[num] == hashed:              Global.logged = true              break          elif content[num] == "":              break          elif content[num] == " ":              break          elif content[num] == "end":              num = 0          if count > 10:              return false          num = num +1      if Global.logged == true:          $LoggedIn.show()          yield(get\_tree().create\_timer(0.5), "timeout")          Global.rege = 2          Global.logopen = false          self.queue\_free()      else:          $Incorrect.show()          yield(get\_tree().create\_timer(2.0), "timeout")          \_ready()      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == "":              return false          if  userline[0] == " ":              return false          if  userline[0] == user:              Global.userInfo = Global.user.new(userline)              return true      return true  func \_on\_Reg\_pressed():      Global.button = true      Global.rege = 1      Global.logopen = false      self.queue\_free()  func \_on\_Quit\_pressed():      get\_tree().quit()  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Log\_pressed() |
| Scene | A picture containing text  Description automatically generated |
| Node tree | A picture containing text  Description automatically generated |

#### Registration v6

|  |  |
| --- | --- |
| What is it? | New button layout, usage of LineEdit instead of TextEdit. Usage of popup dialog boxes instead of labels to tell when the user inputs incorrect data, and if it can any corrections. Change of layout. I also made it so that password have to have at least one number, letter, capital letter, a symbol and be at least 6 letters long; it tells you if you password is missing one of these. |
| Troubles |  |
| Script | extends Control  var alphabet = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"]  var types = [false,false,false,false,true]  func \_ready():      $Max.hide()      $Back.disabled = false      $Quit.disabled = false      $Reg.disabled = false      $Userbox.editable = true      $Passbox.editable = true      $ConPassbox.editable = true      Global.cleanup()      var file = File.new()      var count = 0      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var arrcontent = content.split("\n")      for n in len(arrcontent):          count += 1      if count >= 11:          $Max.show()          $Reg.hide()          $Userbox.hide()          $Passbox.hide()          $ConPassbox.hide()      pass  func searchUser(user):      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              wrong(1)              return false          if  userline[0] == "":              return true          if  userline[0] == " ":              return true      return true  func letterCheck(list):      var flag = false      for i in range(0, 26):          if list.has(alphabet[i]) == true:              flag = true              break      types[1] = flag    func symbolCheck(list):      var listSymbols = ["`","¬","!","£","$","%","^","&","\*","(",")","\_","-","+","=","{","[","}","]",";","@","~","#","<",",",">",".","?","/",":","|"]      var flag = false      for i in len(listSymbols)-1:          if list.has(listSymbols[i].to\_upper()) == true:              flag = true              break      types[3] = flag  func capLetterCheck(list):      var flag = false      for i in len(alphabet)-1:          if list.has(alphabet[i].to\_upper()) == true:              flag = true              break      types[2] = flag  func number(list):      var flag = false      for i in range(0,10):          if list.has(str(i)) == true:              flag = true              break      types[0] = flag  func passCheck(passw):      types[0] = false      types[1] = false      types[2] = false      types[3] = false      types[4] = true      var list =[]      for g in len(passw):          list.append(str(passw[g]))          if passw[g] == "'":              wrong(2)              return false          elif passw[g] == '"':              wrong(2)              return false          elif passw[g] == ',':              wrong(2)              return false      number(list)      letterCheck(list)      capLetterCheck(list)      symbolCheck(list)      if len(list) <= 5:          types[4] = false      for i in len(types):          if types[i] == false:              wrong(5)              return false      return true  func addLogin(user, passw):      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var file2 = File.new()      file2.open("res://userData.dat", file.WRITE)      var add = content+"\n"+user +",0,20,10,2,10,1,0.501187,0,0,0,0,0,3,1,1,0,1"      file2.store\_line(add)      file2.close()      var loco = Global.hashedLocation(user)      var hashed = passw.sha256\_text()      var file3 = File.new()      file3.open("res://HashPass.dat", file.READ)      var content2 = file3.get\_as\_text()      content2 = content2.split("\n", true)      file3.close()      var file4 = File.new()      file4.open("res://HashPass.dat", file.WRITE)      content2[loco] = hashed      for i in len(content2)-1:          file4.store\_line(str(content2[i]))      file4.close()  func wrong(type):      if type == 1:          $Wrong.set\_text("Username already in use")          $Wrong.popup()      elif type == 2:          var text = 'Password is illegal, cannot have the characters " or ' + "'"          $Wrong.set\_text(text)          $Wrong.popup()      elif type == 3:          $Wrong.set\_text("Passwords do not match")          $Wrong.popup()      elif type == 4:          var text = 'Username is illegal, cannot have the characters " or' + "' or , "          $Wrong.set\_text(text)          $Wrong.popup()      elif type == 5:          var text = "Password is not complex enough, must have at least: "          if types[0] == false:              text = text + "a number, "          if types[1] == false:              text = text + "a letter, "          if types[2] == false:              text = text + "a capital letter, "          if types[3] == false:              text = text + "a symbol, "          if types[4] == false:              text = text + "at least 6 characters long, "          $Wrong.set\_text(text)          $Wrong.popup()  func \_on\_Wrong\_popup\_hide():      \_on\_Wrong\_confirmed()  func \_on\_Wrong\_confirmed():      yield(get\_tree().create\_timer(0.5), "timeout")      \_ready()  func userCheck(userName):      var list = []      for g in len(userName):          list.append(str(userName[g]))      for i in len(list)-1:          if ord(list[i]) == ord("'"):              wrong(4)              return false          if ord(list[i]) == ord('"'):              wrong(4)              return false          if ord(list[i]) == ord(","):              wrong(4)              return false      if len(list) < 3:          wrong(4)          return false      else:          return true  func \_on\_Reg\_pressed():      $Back.disabled = true      $Quit.disabled = true      $Reg.disabled = true      $Userbox.editable = false      $Passbox.editable = false      $ConPassbox.editable = false      Global.button = true      var user = $Userbox.text      var goodUser = userCheck(user)      var passw = $Passbox.text      if goodUser == true:          var findu = searchUser(user)          if $Passbox.text == $ConPassbox.text:              if findu == false:                  pass              if findu == true:                  var findp = passCheck(passw)                  if findp == false:                      pass                  if findp == true:                      addLogin(user,passw)                      Global.cleanup()                      Global.rege = 0                      Global.logopen = false                      self.queue\_free()          else:              wrong(3)              pass  func \_on\_Back\_pressed():      Global.button = true      Global.rege = 0      Global.logopen = false      self.queue\_free()  func \_on\_Quit\_pressed():      Global.button = true      get\_tree().quit() |
| Scene |  |
| Node tree |  |

#### Menu v6

|  |  |
| --- | --- |
| What is it? | Changed the colour and layout of the buttons, title and a confirmation popup if logging out or quitting. |
| Troubles |  |
| Script | extends Control  var type = -1  func \_ready():      $Money.text = "Total Cash:" + str(Global.userInfo.getCash())  func \_on\_Logout\_pressed():      Global.button = true      type = 1      stop()  func \_on\_Quit\_pressed():      Global.button = true      type = 0      stop()  func stop():      if type == 0:          $Change.set\_text("Press ok to confirm you are closing the game.")          $Change.popup()      if type == 1:          $Change.set\_text("Press ok to confirm you are logging out.")          $Change.popup()  func \_on\_Change\_confirmed():      Global.button = true      if type == 0:          get\_tree().quit()      if type == 1:          Global.logged = false          Global.rege = 0          Global.logopen = false          self.queue\_free()  func \_on\_Shop\_pressed():      Global.button = true      Global.rege = 3      Global.logopen = false      self.queue\_free()  func \_on\_Game\_pressed():      Global.button = true      Global.rege = 7      Global.logopen = false      self.queue\_free()  func \_on\_Settings\_pressed():      Global.button = true      Global.rege = 5      Global.logopen = false      self.queue\_free()  func \_on\_Controls\_pressed():      Global.button = true      Global.rege = 8      Global.logopen = false      self.queue\_free()    func \_on\_Credits\_pressed():      Global.button = true      Global.rege = 9      Global.logopen = false      self.queue\_free()  func \_on\_Scores\_pressed():      Global.button = true      Global.rege = 4      Global.logopen = false      self.queue\_free()  func \_on\_Lore\_pressed():      Global.button = true      Global.rege = 10      Global.logopen = false      self.queue\_free() |
| Scene |  |

#### Settings v6

|  |  |
| --- | --- |
| What is it? | Same as last time but the added a way to toggle Fullscreen, change the tank’s colour, added button sound effect, some data on the resetting function also changed as I start using class system, added a popup when people click the reset button, I also changed how the volume and sensitivity label update, change of layout, and added away to change hook keybind. |
| Troubles |  |
| Script | extends Control  func \_ready():      $Sens.value = Global.userInfo.getSens()      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userInfo.getSens())      if Global.userInfo.getRotate() == 3:          $Sens.hide()      else:          $Sens.show()      var num = db2linear(Global.userInfo.getVolume())      $Vol.value = num      $Vol/Label.text = "Volume is currently:" + str(int(num\*100))      $HUD.add\_item("Change on how you see the HUD in Game")      $HUD.set\_item\_disabled(0, true)      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")      $Movement.add\_item("Change on how you rotate player")      $Movement.set\_item\_disabled(0, true)      $Movement.add\_item("Left and Right arrow keys")      $Movement.add\_item("Using the A and D keys")      $Movement.add\_item("Using the Mouse location")      $Fire.add\_item("Change on how you fire")      $Fire.set\_item\_disabled(0, true)      $Fire.add\_item("The F key")      $Fire.add\_item("The Space bar")      $Fire.add\_item("The R key")      $Fire.add\_item("Using a left click")      $Colour.add\_item("Change Tank colour")      $Colour.set\_item\_disabled(0, true)      $Colour.add\_item("Black")      $Colour.add\_item("Grey")      $Colour.add\_item("White")      $Colour.add\_item("Purple")      $Colour.add\_item("Brown")      $Colour.add\_item("Orange")      $Colour.add\_item("Red")      $Colour.add\_item("Green")      $Colour.add\_item("Blue")      $Hook.add\_item("Change on how you fire hook")      $Hook.set\_item\_disabled(0, true)      $Hook.add\_item("The H key")      $Hook.add\_item("Shift key")      $Hook.add\_item("Using the Right click")  func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userInfo.setSens(value)      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userInfo.getSens())  func \_on\_Vol\_value\_changed(num):      $Vol/Label.text = "Volume is currently:" + str(int(num\*100))      num = linear2db(num)      Global.userInfo.setVolume(num)  func \_on\_Delete\_pressed():      Global.button = true      Global.rege = 6      Global.logopen = false      self.queue\_free()    func \_on\_Reset\_pressed():      $Res.popup()  func \_on\_Res\_confirmed():      Global.button = true      Global.userInfo.setCash("0")      Global.userInfo.setMaxShell("20")      Global.userInfo.setMaxHP("10")      Global.userInfo.setkitHP("2")      Global.userInfo.setkitAmmo("10")      Global.userInfo.setSens("1")      Global.userInfo.setVolume("0.501187")      Global.userInfo.setH1("0")      Global.userInfo.setH2("0")      Global.userInfo.setH3("0")      Global.userInfo.setH4("0")      Global.userInfo.setH5("0")      Global.userInfo.setHud("3")      Global.userInfo.setRotate("1")      Global.userInfo.setFire("1")      Global.userInfo.setColour("0")      Global.userInfo.setHook("1")      Global.save\_changes()      \_on\_Menu\_pressed()  func \_on\_HUD\_item\_selected(type):      Global.button = true      Global.userInfo.setHud(type)  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Menu\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Menu\_pressed()  func \_on\_Menu\_pressed():      Global.button = true      Global.save\_changes()      Global.rege = 2      Global.logopen = false      self.queue\_free()  func \_on\_Movement\_item\_selected(type):      Global.button = true      if type == 3:          $Sens.hide()      else:          $Sens.show()      Global.userInfo.setRotate(type)  func \_on\_Fire\_item\_selected(type):      Global.button = true      Global.userInfo.setFire(type)  func \_on\_Fullscreen\_pressed():      OS.window\_fullscreen = !OS.window\_fullscreen  func \_on\_Colour\_item\_selected(type):      type = type - 1      Global.button = true      Global.userInfo.setColour(type)  func \_on\_Hook\_item\_selected(index):      Global.button = true      Global.userInfo.setHook(index) |
| Scene |  |
| Node tree |  |

#### Delete v5

|  |  |
| --- | --- |
| What is it? | Added a confirmation box and removed spelling error. |
| Troubles |  |
| Script | extends Control  func \_ready():      $Incorrect.hide()  func \_on\_Settings\_pressed():      Global.button = true      Global.rege = 5      Global.logopen = false      self.queue\_free()      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return true      return false  func incorrectLogin():      $Incorrect.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $Incorrect.hide()      $Userbox.text = ""      $Passbox.text = ""  func \_input(event):      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Settings\_pressed()  func \_on\_Delete\_pressed():      $Del.set\_text("Are you sure you want to delete your account")      $Del.popup()  func \_on\_Del\_confirmed():      Global.button = true      var userName = $Userbox.text      var passWord = $Passbox.text      var foundUser = searchUser(userName)      if foundUser == false:          incorrectLogin()          pass      elif userName != Global.userInfo.getName():          incorrectLogin()          pass      else:          var foundPass = realPass(userName, passWord)          if foundPass == -1:              incorrectLogin()              pass          else:              var file = File.new()              file.open("res://userData.dat", file.READ)              var content = file.get\_as\_text()              content = content.split("\n",true)              file.close()              var counter = len(content)-1              for i in len(content):                  var splitContent = content[i].split(",",true)                  if splitContent[0] == userName:                      content[i] = ""                  counter = counter - 1              file.open("res://userData.dat", file.WRITE)              for g in len(content):                  file.store\_line(content[g])              file.close()              Global.cleanup()              file.open("res://HashPass.dat", file.READ)              content = []              content = file.get\_as\_text().split("\n",true)              file.close()              content[foundPass] = "del"              file.open("res://HashPass.dat", file.WRITE)              for i in len(content)-1:                  file.store\_line(str(content[i]))              file.close()              Global.rege = 0              Global.logopen = false  Global.logged = false              self.queue\_free()  func realPass(user ,passw):      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      file.close()      var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          if content[num] == hashed:              return  num          elif content[num] == "":              return -1          elif content[num] == " ":              return -1          elif content[num] == "end":              num = 0          if count > 10:              return -1          num = num +1 |
| Scene |  |
| Node tree |  |

#### Controls v3

|  |  |
| --- | --- |
| What is it? | Added a text to speech of the text there at an end-user’s request, added the bit for toggling Fullscreen with m key and added the bit for the new Hook call commands. |
| Troubles |  |
| Script | extends Control  func \_ready():      var movement = ""      var shootS = ""      var hook = ""      if Global.userInfo.getRotate() == 1:          movement = "When the Left and Right Arrow Keys are pressed."              $Lr.play()      elif Global.userInfo.getRotate() == 2:          movement = "When the A and D keys are pressed."              $Ad.play()      else:          movement = "Using the cursur location."              $Mouse.play()      if Global.userInfo.getFire() == 3:          shootS = "When the R key is pressed."      elif Global.userInfo.getFire() == 2:          shootS = "When the Space bar is pressed."      elif Global.userInfo.getFire() == 1 :          shootS = "When the F key is pressed."      else:          shootS = "When the left mouse button is pressed"      if Global.userInfo.getHook() == 1:          hook = "When H key is pressed."      elif Global.userInfo.getHook() == 2:          hook = "When the Shift key is pressed."      else:          hook = "Using the right mouse button."      var text = " Movement of player is: "+ movement      text = text + "\n How to shoot the Shell: "+ shootS      text = text + "\n How fire the hook: " + hook      text = text + "\n Press 1 in game for Normal Shell, takes 1 ammo."      text = text + "\n Press 2 in game for Incendiary shell, takes 2 ammo."      text = text + "\n Press 3 in game for Large normal shell, takes 3 ammo."      text = text + "\n To toggle full screen press the m key."      $Stuff.text = text  func \_Rotate\_Finished():      if Global.userInfo.getFire() == 3:          $Rk.play()      elif Global.userInfo.getFire() == 2:          $Space.play()      elif Global.userInfo.getFire() == 1 :          $Fk.play()      else:          $ClickL.play()  func \_Fire\_Finished():      if Global.userInfo.getHook() == 1:          $Hk.play()      elif Global.userInfo.getHook() == 2:          $Shift.play()      else:          $ClickR.play()  func \_Hook\_Finished():      $Shell.play()  func \_on\_Shell\_finished():      $Screen.play()    func \_on\_Menu\_pressed():      Global.button = true      Global.rege = 2      Global.logopen = false      self.queue\_free()  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Menu\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Menu\_pressed() |
| Scene | Graphical user interface  Description automatically generated |
| Node tree | |  | | --- | | Text  Description automatically generated with medium confidence | | Text  Description automatically generated with medium confidence | |

#### Lore v2

|  |  |
| --- | --- |
| What is it? | It is now a video with its own music now that explains the history of the player, it has a skip, back to menu button and replay button. |
| Troubles | Had to learn how to use the video node. |
| Script | extends Control  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Back\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Back\_pressed()  func \_ready():      $Replay.hide()      $Skip.show()      $Fallen.hide()      $Finished.hide()      $VideoPlayer.show()      $VideoPlayer.play()  func \_on\_VideoPlayer\_finished():      $VideoPlayer.hide()      $Skip.hide()      $Finished.show()      $Fallen.show()      $Replay.show()  func \_on\_Replay\_pressed():      \_ready()  func \_on\_Replay\_mouse\_entered():      $Replay.modulate = Color(0,0,0)    func \_on\_Replay\_mouse\_exited():      $Replay.modulate = Color(1,1,1)  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      Global.button = true      self.queue\_free()  func \_on\_Skip\_pressed():      $VideoPlayer.stop()      \_on\_VideoPlayer\_finished()  func \_on\_Back\_mouse\_entered():      $Back.modulate = Color(0,0,0)  func \_on\_Back\_mouse\_exited():      $Back.modulate = Color(1,1,1)  func \_on\_Skip\_mouse\_entered():      $Skip.modulate = Color(0,0,0)  func \_on\_Skip\_mouse\_exited():      $Skip.modulate = Color(1,1,1) |
| Scene | |  | | --- | | A picture containing text, items, sale, several  Description automatically generated | | A picture containing map  Description automatically generated | | A person with a beard  Description automatically generated with low confidence | | A screenshot of a video game  Description automatically generated with low confidence | | A picture containing text, water, sky, outdoor  Description automatically generated | | A basket of bread  Description automatically generated with low confidence | | Map  Description automatically generated | |
| Node tree | Graphical user interface, application  Description automatically generated |

#### Shop v5

|  |  |
| --- | --- |
| What is it? | Fixed the spelling error, “ammount” to “amount” and stopped using the process function. I also shorten the time you see the “NotEnough” node and switched everything to the class system of storing and updating data. |
| Troubles |  |
| Script | |
| extends Control  onready var newHP  onready var newHP\_cost  onready var newAmmo  onready var newAmmo\_cost  onready var newMed  onready var newMed\_cost  onready var newAmmok  onready var newAmmok\_cost  func \_ready():      newHP = Global.userInfo.getMaxHP()+2      newHP\_cost = int(newHP\*2)      newAmmo = Global.userInfo.getMaxShell()+3      newAmmo\_cost = int(newAmmo\*1.5)      newMed = Global.userInfo.getkitHP()+1      newMed\_cost = int(newMed\*2)      newAmmok = Global.userInfo.getkitAmmo()+7      newAmmok\_cost = int(newAmmok\*1.5)      $Money.text = "Total Cash:" + str(Global.userInfo.getCash())      var hptext = "Upgrade Max HP Currently:" + str(Global.userInfo.getMaxHP()) +", upgraded:"+ str(newHP)+ ", upgrade cost:" + str(newHP\_cost)      $Hp.text = str(hptext)      var ammotext = "Upgrade Max Ammo, Currently:" + str(Global.userInfo.getMaxShell()) + ", upgraded:"+ str(newAmmo)+ ", upgrade cost:" + str(newAmmo\_cost)      $Ammo.text = str(ammotext)      var med =  "Upgrade Medkit regen amount, Currently:" + str(Global.userInfo.getkitHP()) +", upgraded:"+str(newMed)+",upgrade cost:" + str(newMed\_cost)      $Med.text = str(med)      var kit = "Upgrade Ammokit regen amount, Currently:" + str(Global.userInfo.getkitAmmo()) + ", upgraded:" + str(newAmmok) +",upgrade cost:" + str(newAmmok\_cost)      $Kit.text = str(kit)  func change(cost):      Global.userInfo.setCash(Global.userInfo.getCash()-cost)      Global.save\_changes()      \_ready()      pass  func check(cost):      if Global.userInfo.getCash() >= cost:          return true      else:          return false  func notEnough():      $NotEnough.show()      yield(get\_tree().create\_timer(1.0), "timeout")      $NotEnough.hide()      pass  func \_on\_AmmoMax\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newAmmo\_cost)      if enoughCash == true:          Global.userInfo.setMaxShell(Global.userInfo.getMaxShell()+3)          change(newAmmo\_cost)      else:          notEnough()      pass  func \_on\_Ammokit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newAmmok\_cost)      if enoughCash == true:          Global.userInfo.setkitAmmo(Global.userInfo.getkitAmmo()+7)          change(newAmmok\_cost)      else:          notEnough()      pass  func \_on\_HP\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newHP\_cost)      if enoughCash == true:          Global.userInfo.setMaxHP(Global.userInfo.getMaxHP()+2)          change(newHP\_cost)      else:          notEnough()      pass  func \_on\_MedKit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newMed\_cost)      if enoughCash == true:          Global.userInfo.setkitHP(Global.userInfo.getkitHP()+1)          change(newMed\_cost)      else:          notEnough()      pass  func \_on\_Menu\_pressed():      Global.button = true      Global.rege = 2      Global.logopen = false      self.queue\_free()  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Menu\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Menu\_pressed() | |
| Scene |  |
| Node tree |  |

#### Scores v6

|  |  |
| --- | --- |
| What is it? | Just shifted everything to use the class system. |
| Troubles |  |
| Script | extends Control  func \_ready():      var allScores =[]      $PersonalScores.text = "1. "+Global.userInfo.getH1()+"\n"+"2. "+Global.userInfo.getH2()+"\n"+"3. "+Global.userInfo.getH3()+"\n"+"4. "+Global.userInfo.getH4()+"\n"+"5. "+Global.userInfo.getH5()+"\n"      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      for i in len(content)-1:          var line = content[i]          line = line.split(",", true)          var add1 = line[0]+". "+line[8]          var add2 = line[0]+". "+line[9]          var add3 = line[0]+". "+line[10]          var add4 = line[0]+". "+line[11]          var add5 = line[0]+". "+line[12]          allScores.append(add1)          allScores.append(add2)          allScores.append(add3)          allScores.append(add4)          allScores.append(add5)      for i in range(allScores.size()-1, -1, -1):          for j in range(1,i+1,1):              var notes = allScores[j].split(". ", true)              var prevnotes = allScores[j-1].split(". ", true)              if int(notes[1]) > int(prevnotes[1]):                  var temp = allScores[j-1]                  allScores[j-1] = allScores[j]                  allScores[j] = temp      var scoresOrdered = ""      for i in len(allScores):          scoresOrdered = scoresOrdered + allScores[i] + "\n"      $HighScores.text = scoresOrdered  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Back\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Back\_pressed()    func \_on\_Back\_pressed():      Global.button = true      Global.rege = 2      Global.logopen = false      self.queue\_free() |
| Scene | A screenshot of a computer  Description automatically generated with low confidence |

#### Game v6

|  |  |
| --- | --- |
| What is it? | Same as last time but now it can launch the “BossZombie” every 60 seconds. Also made a collisions square so the boss cannot leave the screen. |
| Troubles |  |
| Script | extends Node2D  const HUD = preload("res://Scenes/HUD.tscn")  const Player = preload("res://Scenes/Player.tscn")  const Zombie = preload("res://Scenes/Zombie.tscn")  const BossZombie = preload("res://Scenes/BossZombie.tscn")  const AmmoKit = preload("res://Scenes/AmmoKit.tscn")  const MedKit = preload("res://Scenes/MedKit.tscn")  var launchZ = 0  var launchA = 0  var launchM = 0  var num = 1.0  var count =1.0  export (float) var time = 0  func waitz():      yield(get\_tree().create\_timer(1.5), "timeout")      launchZ = 0  func waitm():      yield(get\_tree().create\_timer(20.0), "timeout")      launchM = 0  func waita():      yield(get\_tree().create\_timer(20.0), "timeout")      launchA = 0  func \_ready():      get\_tree().paused = false      Global.pause = 0      Global.game = 1      Global.newGame = 0      Global.ticks = 0      Global.zombieNum = 0      Global.hook = 0      Global.aKit = 0      Global.mKit = 0      Global.boss = false      Global.typeShot = 1      self.add\_child(Player.instance())      self.add\_child(HUD.instance())      var loop = true      while loop == true:          yield(get\_tree().create\_timer(3.0), "timeout")          if Global.pause == 0:              if Global.boss == false:                  Global.ticks = Global.ticks + 0.5  func \_process(\_delta):      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      elif Global.newGame == 1:          Global.logopen = false          self.queue\_free()      elif Global.pause == 1:          pass      else:          if Global.zombieNum < 3:              if Global.boss == false:                  self.add\_child(Zombie.instance())                  Global.zombieNum =Global.zombieNum  +  1          if launchZ == 0:              if Global.zombieNum < 20:                  if Global.boss == false:                      self.add\_child(Zombie.instance())                      Global.zombieNum =Global.zombieNum  +  1                      launchZ = 1                      waitz()          if launchM == 0:              if Global.mKit == 0:                  if Global.boss == false:                      if Global.hp < Global.userInfo.getMaxHP():                          self.add\_child(MedKit.instance())                          Global.mKit = 1                          launchM = 1                          waitm()          if launchA == 0:              if Global.aKit == 0:                  if Global.boss == false:                      if Global.ammo < Global.userInfo.getMaxShell():                          self.add\_child(AmmoKit.instance())                          Global.aKit = 1                          launchA = 1                          waita()          num = Global.ticks/10          if num == count:              if Global.boss == false:                  Global.boss = true                  count = count +1.0                  self.add\_child(BossZombie.instance())                  Global.ticks = Global.ticks + 0.5 |
| Scene |  |
| Node tree |  |

#### HUD v7

|  |  |
| --- | --- |
| What is it? | Same as last time but changes the ammo bar colour to blue and shifted the ammo type indicator to the left side of the ammo bar. Shifted code to the class system also. Added in game and game over music. Changed layout also, so you can actually see the tank when it is game over. |
| Troubles |  |
| Script | extends Control  var Hudtype = Global.userInfo.getHud()  var scores = []  var getCash = Global.cash  func \_ready():      $ProHUDHP.set\_max(Global.userInfo.getMaxHP())      $ProHUDAmmo.set\_max(Global.userInfo.getMaxShell())      Global.hp = int(Global.userInfo.getMaxHP())      Global.ammo = int(Global.userInfo.getMaxShell())      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      $Over.hide()          $Play.play()      scores.append(int(Global.userInfo.getH1()))      scores.append(int(Global.userInfo.getH2()))      scores.append(int(Global.userInfo.getH3()))      scores.append(int(Global.userInfo.getH4()))      scores.append(int(Global.userInfo.getH5()))      pass    func \_on\_Pause\_pressed():      Global.button = true      $Pause.hide()      Global.pause = 1      $Play.stream\_paused = true      get\_tree().paused = true      $Unpause.show()      $Quit.show()  func \_on\_Unpause\_pressed():      Global.button = true      $Pause.show()      Global.pause = 0      $Play.stream\_paused = false      get\_tree().paused = false      $Unpause.hide()      $Quit.hide()  func \_on\_Quit\_pressed():      Global.button = true      Global.game = 0      Global.pause = 0      get\_tree().paused = false  func \_process(\_delta):      if Global.cash>0:          $Data.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      else:          $Data.text = " Score: "+str(Global.score)      if Global.typeShot == 1:          $TypePic.frame = 0          $TypeText.text = "Normal"      if Global.typeShot == 2:          $TypePic.frame = 1          $TypeText.text = "Incendiary"      if Global.typeShot == 3:          $TypePic.frame = 2          $TypeText.text = "Large"      if Hudtype == 1:          $HUD.text = " HP: " + str(Global.hp) + " of "+str(Global.userInfo.getMaxHP()) +"\nAmmo: "+ str(Global.ammo) + " of "+ str(Global.userInfo.getMaxShell())          $ProHUDHP.hide()          $ProHUDAmmo.hide()      elif Hudtype == 2:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      else:          $HUD.text = " HP: " + str(Global.hp) + " of "+str(Global.userInfo.getMaxHP()) +"\n Ammo: "+ str(Global.ammo) + " of "+ str(Global.userInfo.getMaxShell())          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)      if Global.hp <= 0:          Global.hp = 0          $Play.playing = false          if Global.pause == 0:              $Play.playing = false                  $Death.play()              get\_tree().paused = true              $Pause.hide()              changes()              Global.pause = 1      else:          if Global.boss == true:              $Play.stream\_paused = true          else:              if Global.pause != 1:                  $Play.stream\_paused = false    func \_on\_Menu\_pressed():      Global.button = true      Global.game = 0      get\_tree().paused = false      self.queue\_free()  func \_on\_Restart\_pressed():      Global.button = true      Global.newGame = 1      get\_tree().paused = false      self.queue\_free()  func changes():      if Global.pause == 0:          Global.pause = 1          $Over.show()          var change = false          scores.append(Global.score)          Global.userInfo.setCash(Global.userInfo.getCash()+Global.cash)          Global.cash = 0          $Over/TotalCash.text = "Total cash is " + str(Global.userInfo.getCash())          for i in range(scores.size()-1, -1, -1):              for j in range(1,i+1,1):                  var notes = scores[j]                  var prevnotes = scores[j-1]                  if notes > prevnotes:                      var temp = scores[j-1]                      scores[j-1] = scores[j]                      scores[j] = temp                      change = true          for i in len(scores):              $Over/Scores.text = $Over/Scores.text + str(scores[i]) + "\n"          if change == true:              Global.userInfo.setH1(str(scores[0]))              Global.userInfo.setH2(str(scores[1]))              Global.userInfo.setH3(str(scores[2]))              Global.userInfo.setH4(str(scores[3]))              Global.userInfo.setH5(str(scores[4]))          Global.save\_changes()  func \_input(event):      if event.is\_action\_pressed("ui\_1"):          Global.typeShot = 1      if event.is\_action\_pressed("ui\_2"):          Global.typeShot = 2      if event.is\_action\_pressed("ui\_3"):          Global.typeShot = 3      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Pause\_pressed()  func \_on\_Play\_finished():      $Play.play()  func \_on\_Death\_finished():      $End.play()  func \_on\_End\_finished():      $End.play() |
| Scene | |  | | --- | |  | |  | |  | |  | |

#### Player v6

|  |  |
| --- | --- |
| What is it? | I now allow the user to change tank colour (Black, Grey, White, Purple, Brown, Orange, Red, Green and Blue) of those they are split into 3 groups that have distinct shape, so I now have multiple collision nodes and activate the one that corresponds to the correct colour. There is also now a scope that comes out when the tank is rotating in any way except by mouse location. The scope also changes colour to correspond to the tank when not in global mouse position mode. Added the bits to allow to change keybinds for the hook. The scope now is coloured to the same colour of the tank and it will be at mouse position if the player controls by movement by mouse position. |
| Troubles |  |
| Script | extends KinematicBody2D  const Shell = preload("res://Scenes/Shell.tscn")  const Hook = preload("res://Scenes/Hook.tscn")  const ShellFire = preload("res://Scenes/ShellFire.tscn")  const ShellLarge = preload("res://Scenes/ShellLarge.tscn")  export (float) var rotation\_speed  var rotation\_dir = 0  var tleft = ""  var tright = ""  var tfire = ""  var thook = ""  var mouseLook = false  var target = Vector2()  var collision  onready var main = get\_tree().current\_scene  func \_ready():      rotation\_speed = Global.userInfo.getSens()\*5      $Blue.disabled = true      $Red.disabled = true      $Green.disabled = true      $Tank\_dark.animation = "default"      $Tank\_dark.frame = Global.userInfo.getColour()      $Scope.frame = Global.userInfo.getColour()      $Tank\_dark.playing = false      $Tank\_dark.show()      $Scope.show()      if Global.userInfo.getColour() == 0:          $Green.disabled = false      elif Global.userInfo.getColour() == 1:          $Blue.disabled = false      elif Global.userInfo.getColour() == 2:          $Blue.disabled = false      elif Global.userInfo.getColour() == 3:          $Green.disabled = false      elif Global.userInfo.getColour() == 4:          $Red.disabled = false      elif Global.userInfo.getColour() == 5:          $Red.disabled = false      elif Global.userInfo.getColour() == 6:          $Red.disabled = false      elif Global.userInfo.getColour() == 7:          $Green.disabled = false      elif Global.userInfo.getColour() == 8:          $Blue.disabled = false      if Global.userInfo.getRotate() == 1:          tleft = "ui\_right"          tright = "ui\_left"      elif Global.userInfo.getRotate() == 2:          tright = "ui\_a"          tleft = "ui\_d"      else:          mouseLook = true      if Global.userInfo.getFire() == 3:          tfire = "ui\_r"      elif Global.userInfo.getFire() == 2:          tfire = "ui\_space"      elif Global.userInfo.getFire() == 1:          tfire = "ui\_f"      else:          tfire = "ui\_left\_click"      if Global.userInfo.getHook() == 1:          thook = "ui\_h"      elif Global.userInfo.getHook() == 2:          thook = "ui\_shift"      else:          thook = "ui\_right\_click"  func \_process(delta):      rotation\_dir = 0      if Global.hp <= 0:          if Global.pause == 0:              $Scope.hide()              $Tank\_dark.animation = "blow"              $Tank\_dark.frame = 0              $Tank\_dark.playing = true      if mouseLook == false:          if Input.is\_action\_pressed(tleft):              rotation\_dir += 1          if Input.is\_action\_pressed(tright):              rotation\_dir -= 1          self.rotation += rotation\_dir \* rotation\_speed \* delta      else:          if Global.pause == 0:              look\_at(get\_global\_mouse\_position())              $Scope.position = self.get\_local\_mouse\_position()              Input.set\_mouse\_mode(Input.MOUSE\_MODE\_HIDDEN)          else:              Input.set\_mouse\_mode(Input.MOUSE\_MODE\_VISIBLE)  func \_input(event):      if event.is\_action\_pressed(tfire):          if Global.pause == 0:              if Global.firing == 0:                  if Global.typeShot == 1:                      if Global.ammo > 0:                          Global.firing = 1                          var shell = Shell.instance()                          main.add\_child(shell)                          shell.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -1                          $SFXFire.play()                          yield(get\_tree().create\_timer(0.3), "timeout")                          Global.firing = 0                  elif Global.typeShot == 2:                      if Global.ammo > 1:                          Global.firing = 1                          var shellfire = ShellFire.instance()                          main.add\_child(shellfire)                          shellfire.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -2                          $SFXFire.play()                          yield(get\_tree().create\_timer(1.0), "timeout")                          Global.firing = 0                  elif Global.typeShot == 3:                      if Global.ammo > 2:                          Global.firing = 1                          var shelllarge = ShellLarge.instance()                          main.add\_child(shelllarge)                          shelllarge.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -3                          $SFXFire.play()                          yield(get\_tree().create\_timer(0.6), "timeout")                          Global.firing = 0      if event.is\_action\_pressed(thook):          if Global.pause == 0:              if Global.hook == 0:                  Global.hook = 1                  var hook = Hook.instance()                  main.add\_child(hook)                  hook.global\_transform = $Muzzle.global\_transform  func \_on\_Tank\_dark\_animation\_finished():      $Tank\_dark.hide() |
| Scene |  |
| Node tree |  |
| Colours |  |

#### Shell v5

|  |  |
| --- | --- |
| What is it? | A cleaned-up version of the previous |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = false  var playing = false  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.show()      $Sprite.hide()  func \_physics\_process(delta):      velocity += transform.y \* speed      if Global.pause == 1:          pass      elif hit == false:          collision = move\_and\_collide(velocity \* delta)          if collision:              if str(collision.collider.name) == "Zombie":                  hit = true      else:          if playing == false:              playing = true              $Bullet.hide()              $Sprite.frame = 0              $Sprite.show()              $Sprite.playing = true              $CollisionShape2D.scale.x = 4              $CollisionShape2D.scale.y = 4  func \_on\_Sprite\_animation\_finished():      self.queue\_free()  func \_on\_Out\_screen\_exited():      self.queue\_free() |

#### ShellFire v2

|  |  |
| --- | --- |
| What is it? | Decreased the size and length of explosion as a game balance and it was to overpower, I also cleaned it up |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = false  var playing = false  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.show()      $Sprite.hide()  func \_physics\_process(delta):      velocity += transform.y \* speed      if Global.pause == 1:          pass      elif hit == false:          collision = move\_and\_collide(velocity \* delta)          if collision:              if str(collision.collider.name) == "Zombie":                  hit = true      else:          if playing == false:              playing = true              $Bullet.hide()              $Sprite.frame = 0              $Sprite.show()              $Sprite.playing = true  func \_on\_Sprite\_animation\_finished():      self.queue\_free()  func \_on\_Out\_screen\_exited():      self.queue\_free() |

#### ShellLarge v2

|  |  |
| --- | --- |
| What is it? | Same as previous but cleaned it up |
| Troubles |  |
| Script | extends KinematicBody2D  export (int) var speed = -100  var hit = false  var playing = false  var velocity = Vector2()  var collision  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.show()      $Sprite.hide()  func \_physics\_process(delta):      velocity += transform.y \* speed      if Global.pause == 1:          pass      elif hit == false:          collision = move\_and\_collide(velocity \* delta)          if collision:              if str(collision.collider.name) == "Zombie":                  hit = true      else:          if playing == false:              playing = true              $Bullet.hide()              $Sprite.frame = 0              $Sprite.show()              $Sprite.playing = true              $CollisionShape2D.scale.x = 8              $CollisionShape2D.scale.y = 8  func \_on\_Sprite\_animation\_finished():      self.queue\_free()  func \_on\_Out\_screen\_exited():      self.queue\_free() |

#### Zombie v6

|  |  |
| --- | --- |
| What is it? | I moved the tank’s location so have to change code for the zombies, so they know where to look at. Also removed the old volume control. |
| Troubles |  |
| Script | extends KinematicBody2D  var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  var hit = false  var target = Vector2(990, 600)  func \_ready():      rng.randomize()      if where == 1:          if side == 0:              var along =  rng.randi\_range(0,1080)              position.y = along          else:              position.x = 1980              var along =  rng.randi\_range(0,1080)              position.y = along      else:          if side == 0:              var along =  rng.randi\_range(0,1980)              position.x = along          else:              position.y = 1080              var along =  rng.randi\_range(0,1980)              position.x = along      $Sprite.modulate = Color(rng.randf\_range(0.5,1),rng.randf\_range(0.5,1),rng.randf\_range(0.5,1))  func \_physics\_process(\_delta):      look\_at(target)      var inc = Global.ticks/5      velocity = position.direction\_to(target)\*inc      if Global.game == 0:          self.queue\_free()      if Global.hp <= 0:          self.queue\_free()      if Global.pause == 1:          $Sprite.playing= false          if Global.newGame == 1:              self.queue\_free()          pass      elif position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)      if Global.pause == 0:          $Sprite.playing= true          if collision:              var item = str(collision.collider.name)              item = item.replace("@","")              item = item.replace("1","")              item = item.replace("2","")              item = item.replace("3","")              item = item.replace("4","")              item = item.replace("5","")              item = item.replace("6","")              item = item.replace("7","")              item = item.replace("8","")              item = item.replace("9","")              item = item.replace("0","")              if hit ==false:                  if item == "Shell":                      hit = true                      Global.score = Global.score + int(Global.ticks)                      Global.cash = Global.cash +1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXDie.play()                      yield(get\_tree().create\_timer(0.45), "timeout")                      self.queue\_free()                  elif item == "Tank":                      hit = true                      Global.hp = Global.hp -1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXEat.play()                      yield(get\_tree().create\_timer(1.3), "timeout")                      self.queue\_free() |

#### Hook v3

|  |  |
| --- | --- |
| What is it? | Change of sprite and updating of script to have the class system. |
| Script | extends KinematicBody2D  export (int) var speed = -100  var velocity = Vector2()  var collision  func \_process(delta):      velocity += transform.y \* speed      collision = move\_and\_collide(velocity \* delta)      if collision:          if str(collision.collider.name) == "MedKit":              Global.hp = Global.hp + Global.userInfo.getkitHP()              if Global.hp > Global.userInfo.getMaxHP():                  Global.hp = Global.userInfo.getMaxHP()              Global.hook = 0              Global.mKit = 0              self.queue\_free()          if str(collision.collider.name) == "AmmoKit":              Global.ammo = Global.ammo + Global.userInfo.getkitAmmo()              if Global.ammo > Global.userInfo.getMaxShell():                  Global.ammo = Global.userInfo.getMaxShell()              Global.hook = 0              Global.aKit = 0              self.queue\_free()  func \_on\_Out\_screen\_exited():      Global.hook = 0      self.queue\_free() |
| Scene |  |

#### AmmoKit v4

|  |  |
| --- | --- |
| What is it? | Updating of script to have the class system, shorter and added notations. |
| Troubles |  |
| Script | extends KinematicBody2D  var hit =false  var rng = RandomNumberGenerator.new()  func \_ready():      rng.randomize()      self.position.x = rng.randi\_range(100,1880)      self.position.y = rng.randi\_range(100,980)      yield(get\_tree().create\_timer(10.0), "timeout")      Global.aKit = 0      self.queue\_free()  func \_process(\_delta):      if Global.hp <= 0:          self.queue\_free()      if hit == false:          if Global.aKit == 0:              hit = true              self.hide()              $Get.play()              yield(get\_tree().create\_timer(0.5), "timeout")              self.queue\_free() |

#### MedKit v4

|  |  |
| --- | --- |
| What is it? | Updating of script to have the class system. |
| Troubles |  |
| Script | extends KinematicBody2D  var hit =0  var rng = RandomNumberGenerator.new()  func \_ready():      rng.randomize()      self.position.x = rng.randi\_range(100,1880)      self.position.y = rng.randi\_range(100,980)      yield(get\_tree().create\_timer(10.0), "timeout")      Global.mKit = 0      self.queue\_free()  func \_process(\_delta):      if Global.hp <= 0:          self.queue\_free()      if hit == 0:          if Global.mKit == 0:              hit = 1              self.hide()              $Get.play()              yield(get\_tree().create\_timer(0.76), "timeout")              self.queue\_free() |

#### BossZombie v1

|  |  |
| --- | --- |
| What is it? | It will spawn to any side like the common zombie, but do 100 points of hp damage, they also play boss music. There HP will be the current score. |
| Troubles |  |
| Script | extends KinematicBody2D  var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  var hp = Global.score  var yes = true  var target = Vector2(990, 600)  func \_ready():      $ZombieHP.set\_max(hp)            $Battle.play()      rng.randomize()      $Sprite.playing= true      if where == 1:          if side == 0:              position.y = rng.randi\_range(100,980)              position.x = 92          else:              position.x = 1880              position.y = rng.randi\_range(100,980)      else:          if side == 0:              position.x = rng.randi\_range(100,1880)              position.y = 100          else:              position.y = 980              position.x = rng.randi\_range(100,1990)      $Sprite.modulate = Color(rng.randf\_range(0.5,1),rng.randf\_range(0.5,1),rng.randf\_range(0.5,1))  func \_physics\_process(\_delta):      if hp <= 0:          Global.score = Global.score +30 + int(Global.ticks)          Global.cash = Global.cash +30 + int(Global.ticks)          Global.boss = false          self.queue\_free()      look\_at(target)      velocity = position.direction\_to(target)\*0.25      if Global.hp <= 0:          self.queue\_free()      if Global.pause == 1:          $Sprite.playing= false          $Battle.stream\_paused = true          pass      if Global.pause == 0:          $Battle.stream\_paused = false          $Sprite.playing= true          collision = move\_and\_collide(velocity)          $ZombieHP.set\_value(hp)          if collision:              var item = str(collision.collider.name)              if item == "Shell":                  if yes == true:                      yes = false                      hp = hp-1                      yield(get\_tree().create\_timer(0.01), "timeout")                      yes = true              elif item == "Tank":                  Global.hp = Global.hp -100                  self.hide()                  self.position.x =0                  self.position.y =0                  Global.boss = false                  self.queue\_free()  func \_on\_Battle\_finished():          $Battle.play() |
| Scene |  |
| Node tree |  |

#### White box testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Change from previous | Expected | Outcome | Evidence |
| Boss zombie spawn every 60 seconds |  | Will spawn | It spawned |  |
| Boss zombie instantly killing player |  | Game over | Game over |  |
| Death music plays |  | Once game over has occurred the “End” music node will play | It did |  |
| The Game music “Play” will play |  | During the game the “Play” node will play | Played |  |
| During the boss round the boss music will play |  | The “Battle” node will play | It played |  |
| No other being/item will spawn during boss battles |  | The Game scene will not instance any entity during the boss round. | No entities spawned. |  |
| The tank will change if the setting is changed |  | The tank changes | It did | |  | | --- | |  | |  | |
| New registration system works |  | It will tell you how your registration details are wrong |  | |  | | --- | |  | |  | |  | |  | |
| Video plays in lore |  | Video plays in lore | Did it. |  |
| Tank doesn’t rotate when paused |  | Tank stops rotating when paused | It stopped for all of the controls except mouse |  |
| Tank doesn’t rotate when paused when using mouse | Put the mouse rotate code under an if statement so it only works when not paused | Tank stops rotating | It worked |  |
| Check if shop accept using new mechanic |  | Shop shows prices and stuff | It doesn’t work. The ready and process function are not natively called. |  |
| Check if shop accept using new mechanic | Restarted Godot | It stated working |  |  |
| Pop-up work |  | All popups work | Worked | |  | | --- | |  | |  | |  | |  | |  | |
| New keybinds for hook work. |  | Hook fires on H key. | Works |  |
| New keybinds for hook work. |  | Hook fires on shift key. | Works |  |
| New keybinds for hook work. |  | Hook fires right mouse click. | Works |  |
| Controls show new sound effects controls |  | When the hook setting is set on H key it plays node Hk, and display usage of H key for firing hook. | Works |  |
| Controls show new sound effects |  | When the hook setting is set on Shift key it plays node Shift, and display usage of Shift key for firing hook. | Works |  |
| Controls show new sound effects |  | When the hook setting is set on right click it plays node ClickR, and display usage of right click for firing hook. | works |  |
| The new instancing in main.gd working |  | Scene instance like they did. | works |  |

#### Summary

On this iteration I overhauled everything: in main I changed the background system so it will show 1 of six images with 2 tanks firing at a boss zombie, control volume via the volume bus and shortened the code to instance scenes; I changed how data is stored in game from a list to a class; added more usability features to the game by making the text boxes in the registration and login have ghost text; For registration the password is now secure as it makes the person make a complex password, fulfilling the success criteria; the game now uses confirmation boxes when the player is trying to do something important (resetting data in settings, deleting their account in delete, logging out and quitting the game in menu); The setting has been upgraded allowing the player to customize their character and edit keybind for the hook; made a text to speech for the controls section; upgraded the lore scene to have video cutscene; removed the spelling error in the shop scene.

The game scene can now instance a boss, which will its health be the current score. It will also do 100 points of damage to the player. The boss also plays its own boss music, this compliments the normal in-game music played by the HUD, fully fulfilling the success criteria for themed music. The player now has a scope that will point to where the shell will fire, it will even follow the cursor if it is on mouse position mode. As the incendiary shell was to game breaking the size of its explosions was decreased.

### Summary of prototype 3

This is the last prototype and now the final version discounting any game patches or bug fixes sent out later. This prototype focused on finishing of the rest of the success criteria and tying up everything to make the game good

|  |  |  |
| --- | --- | --- |
| Success Criteria | Fulfilled or not | Where |
| People can make up to 10 accounts on each game instance, so people do not make an infinite amount, filling the storage until there is a computer crash. | Fulfilled | [Registration v1](#_Registration_v1) |
| The player can exit the game from the HUD. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can pause the gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| Make sure login works. | Fulfilled | [Login v2](#_Login_v2) |
| Make sure the password system is secure | Fulfilled | [Registration v5](#_Registration_v5) |
| The person can change input settings (key bindings). | Fulfilled | [Settings v2](#_Settings_v2) |
| The person can change volume settings. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change delete all records of their scores. | Fulfilled | [Delete v1](#_Delete_v1) |
| The person can change the sensitivity of movement. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change reset all progress. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can make it full screen. | Fulfilled | [Settings v5](#_Settings_v5) |
| The person can change if they want the HP to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change if they want the ammo to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) |
| The person can change and delete their account. | Fulfilled | [Delete v1](#_Delete_v1)  [Settings v1](#_Settings_v1) |
| The player can see the amount of in-game cash they have in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can see the total in-game cash they have on the menu. | Fulfilled | [Menu v1](#_Menu_v1) |
| The player can be able to see their top 5 scores. | Fulfilled | [Scores v2](#_Scores_v2) |
| The player can be able to see the top 5 scores from all accounts. | Fulfilled | [Scores v2](#_Scores_v2) |
| The player can upgrade themselves in the in-game shop with in-game cash. | Fulfilled | [Shop v1](#_Shop_v1) |
| Themed music must play in certain scenes. | Fulfilled | [Main v6](#_Main_v6) |
| Some lore that is interesting | Fulfilled | [Lore v2](#_Lore_v2) |
| The enemy moves towards the player and face the player. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The player can rotate | Fulfilled | [Player v1](#_Player_v1) |
| The shell travels in the direction fired | Fulfilled | [Shell v1](#_Shell_v1) |
| The hook travels in the direction fired | Fulfilled | [Hook v1](#_Hook_v1) |
| The kits will appear randomly. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| Upon gameplay, the rotation speed is affected by the sensitivity set in the settings. | Fulfilled | [Player v1](#_Player_v1) |
| The player can fire a tank shell upon the assigned key binding. | Fulfilled | [Shell v1](#_Shell_v1)  [Player v1](#_Player_v1) |
| The player can fire a hook upon assigned key binding. | Fulfilled | [Player v1](#_Player_v1)  [Hook v1](#_Hook_v1) |
| The player’s health must decrease if the enemy touches them and then the enemy disappears. | Fulfilled | [HUD v1](#_HUD_v1) |
| The enemy must disappear if touching shell/explosion and increase score and in-game cash. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The enemies speed and damage done to the player must increase linearly during one gameplay and reset after it. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The enemy must appear randomly around the player. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| Tank-repair-kits and ammo kits must appear randomly on the screen. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| Tank-repair-kits and ammo kits must disappear when touching the hook and increase the appropriate bar. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1)  [Hook v1](#_Hook_v1) |
| Tank-repair-kits and ammo kits must disappear in a set amount of time if it is not used. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) |
| The player is defeated when health is 0. | Fulfilled | [HUD v1](#_HUD_v1)  [Zombie v1](#_Zombie_v1) |
| The player can see enemies. | Fulfilled | [Zombie v1](#_Zombie_v1) |
| The player can see the playable character. | Fulfilled | [Player v1](#_Player_v1) |
| The player can interact with the game. | Fulfilled | [Player v1](#_Player_v1) |
| Player can change ammo type | Fulfilled | [HUD v6](#_HUD_v6) |
| Must have at least 2 variants of enemies, normal and boss | Fulfilled | [Game v6](#_Game_v6)  [Zombie v1](#_Zombie_v1)  [BossZombie v1](#_BossZombie_v1) |
| The player can see their ammo and HP levels. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player can see the score they got in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) |
| Must be able to pause and quit the game. | Fulfilled | [HUD v1](#_HUD_v1) |
| The player must be able see the amount of cash they got | Fulfilled | [HUD v3](#_Hud_v3) |

### References for prototype 3

|  |  |
| --- | --- |
| What | Where |
| Audio control | <https://www.gdquest.com/tutorial/godot/audio/volume-slider/> |
| Classes | <https://godotengine.org/qa/9830/handling-classes-in-godot> |
| Video | <https://docs.godotengine.org/en/stable/classes/class_videoplayer.html> |

# Evaluation

## Post development testing

Here I test for robustness and usability of my exported game into a exe file by giving it false inputs, seeing its reaction and how it feels to play.

### Registration and login

|  |  |  |  |
| --- | --- | --- | --- |
| What I am testing | What I expect | What happens | Evidence |
| Test to see if button that changes scene to registration works and back | Changes scene to registration then back to login | Scene changes to registration then back to login | Video: <https://youtu.be/3R96poKCVDA> |
| Test for inputting wrong details into registration scene | It will show a dialog box telling me what happens. | The dialog box opened and told me my input was invalid and game a reason why. | Video: <https://youtu.be/3R96poKCVDA> |
| Test if inputting working registration details to see if the database will update | The database will have a new line with a new user | The database updated |  |
| The new login works | I will login to the new account | The menu is shown of the new account | Video: <https://youtu.be/3R96poKCVDA?t=80> |

### Menu

|  |  |  |  |
| --- | --- | --- | --- |
| What I am testing | What I expect | What happens | Evidence |
| If the shop button is pressed the shop opens | The shop opens | The shop opened | <https://youtu.be/3R96poKCVDA?t=215> |
| If the controls button is pressed the controls scene opens | The controls open | The controls opened | <https://youtu.be/3R96poKCVDA?t=167> |
| If the settings button is pressed and the settings scene opens | The settings open | The settings opened | <https://youtu.be/3R96poKCVDA?t=120> |
| If the credits scene opens if I press the credits button | The credits open | The credits opened | <https://youtu.be/3R96poKCVDA?t=160> |
| If the scores scene opens if I press the scores button | The scores open | The scores opened | <https://youtu.be/3R96poKCVDA?t=193> |
| The game shows the current amount of cash the player has | The correct amount of cash was shown | It was the correct amount of cash | <https://youtu.be/3R96poKCVDA?t=466> |
| If the game closes if press confirm to the quit dialog box | The program will close | The program closed | <https://youtu.be/3R96poKCVDA?t=775> |
| If the lore scene will load if I press the lore button | The lore opens | The lore opened | <https://youtu.be/3R96poKCVDA?t=223> |
| If the game will logout if I confirm the logout dialog box | The game logs out | It logged out | <https://youtu.be/3R96poKCVDA?t=752> |
| The game loop starts if I press the play button | The game loop starts | The game loop started | <https://youtu.be/3R96poKCVDA?t=349> |
| Seeing if you can log out and log back in with a different account | The other accounts logins | The other account logged in | <https://youtu.be/w0SCHp6pzrU?t=105> |

### Shop

|  |  |  |  |
| --- | --- | --- | --- |
| What I am testing | What I expect | What happens | Evidence |
| Seeing what happens when I press the upgrade buttons frequently when I don’t have the cash to upgrade. | The not enough node would appear | The not enough node would appear | <https://youtu.be/w0SCHp6pzrU?t=12> |

### Lore

|  |  |  |  |
| --- | --- | --- | --- |
| What I am testing | What I expect | What happens | Evidence |
| Seeing if the video player node bugs out if the skip and replay button are used repeatedly | The not enough node would appear | The not enough node appeared | <https://youtu.be/w0SCHp6pzrU?t=56> |

### HUD

|  |  |  |  |
| --- | --- | --- | --- |
| What I am testing | What I expect | What happens | Evidence |
| Seeing if the ammo type display does not break if you change the ammo type erratically | The ammo type node shows the correct one | The ammo type node showed the correct one | <https://youtu.be/w0SCHp6pzrU?t=301> |
| Seeing if pausing and unpausing quickly breaks the HUD | The game pauses and resumes correctly | The game paused and resumed well | <https://youtu.be/w0SCHp6pzrU?t=229>  Note that you can also pause the game by pressing the esc key |

### Settings

|  |  |  |  |
| --- | --- | --- | --- |
| What I am testing | What I expect | What happens | Evidence |
| Seeing if I change the volume erratically if it will crash the game | The game will not crash | The game did not crash | <https://youtu.be/w0SCHp6pzrU?t=90> |
| Seeing if I change the sensitivity erratically if it will crash the game | The game will not crash | The game did not crash | <https://youtu.be/w0SCHp6pzrU?t=134> |
| Seeing what happens if I toggle the full screen button repeatedly in quick succession | The game not crashing | The game did not crash | <https://youtu.be/w0SCHp6pzrU?t=82> |
| Seeing what happens if I reset player data repeatedly in quick succession | The game saving the data correctly, so it is used properly | The game did save the data correctly and no errors occurred | <https://youtu.be/w0SCHp6pzrU?t=116> |

### Usability

The testing for usability testing was done by getting the stakeholder Zach Tracy to play the game (he had deafened himself to previous reports as he wanted to play the fished game as if here a new player).

Full video: <https://youtu.be/p-JHLRDjVvo>

|  |  |  |  |
| --- | --- | --- | --- |
| What he is doing | Usability feature | Is it successful | Time stamp |
| He makes an account, he found that it was simple enough to understand, most games have this feature, so he understands it. | The registration of a new account | Partial success, unless you get the password wrong the scene does not tell you password requirements. | <https://youtu.be/p-JHLRDjVvo?t=0> |
| He signs in, simple thing that most people understand. | Logging in | Success, very simple and easy to understand. | <https://youtu.be/p-JHLRDjVvo?t=60> |
| He checks out the lore scene, finds the music good but he found the video was not interesting enough for him, but he rates it as an 7/10 but did not like the ending of it. | Music and lore | The themed music was good, but the lore was a near success, the last frame just needs to be changed. | <https://youtu.be/p-JHLRDjVvo?t=69> |
| He checks out the shop, he liked it. | Shop | He understood it so it was a success | <https://youtu.be/p-JHLRDjVvo?t=89> |
| He checks out the score viewer, found the font good and sleek. | Scores | Success as it was simple and understandable | <https://youtu.be/p-JHLRDjVvo?t=97> |
| He finds out his controls, and did not understand it, (he usually does not usually look at them). | Controls | Success, he usually goes into the deep end when playing game for the first time. | <https://youtu.be/p-JHLRDjVvo?t=110> |
| He changes his settings, to the mouse inputs as it makes the game easier. | Keybind changer | Success, the keybind changed | <https://youtu.be/p-JHLRDjVvo?t=120> |
| He starts playing, found it fun and challenging | Game itself | Success | <https://youtu.be/p-JHLRDjVvo?t=154> |
| He gets his first boss encounter, got angry of dying | Game itself |  | <https://youtu.be/p-JHLRDjVvo?t=217> |
| His player dies, angry | Game itself |  | <https://youtu.be/p-JHLRDjVvo?t=264> |
| He upgrades the max ammo, as he died due to less ammo, he upgraded the max ammo, making the game more fun as he can play longer. | Shop | Success, he was able to upgrade his player | <https://youtu.be/p-JHLRDjVvo?t=270> |
| He plays again, he found it more enjoyable as he understands the game now | Game itself |  | <https://youtu.be/p-JHLRDjVvo?t=272> |
| He gets second boss encounter, he got angry as he does not have enough ammo | Game itself |  | <https://youtu.be/p-JHLRDjVvo?t=331> |
| His player dies, he is frustrated now. | Game itself |  | <https://youtu.be/p-JHLRDjVvo?t=373> |
| He rage-quits | Menu | Success the quit button works well. | <https://youtu.be/p-JHLRDjVvo?t=377> |

Even at the angry ending he found the game quite usable if you understand the main mechanics of most games but is unsuitable for people just starting out gaming. To fix this, for further development the game could incorporate a tutorial for new users.

## Post development Black box testing

Here I send my game to beta testers for them to break it, they then send a bug report when they are done.

### Bug 1

|  |  |
| --- | --- |
| What is the bug | If the login button were pressed when the text boxes were empty the game would crash |
| How the bug effects the game | It would be a hindrance to players. |
| Issue | The scanner for empty boxes was after the part for looking accessing data. |
| Bugged code | Text  Description automatically generated |
| Reworked code | A screenshot of a computer  Description automatically generated with medium confidence |
| Does the change work | Yes |

### Bug 2

|  |  |
| --- | --- |
| What is the bug | The upgrade for ammo max is showing an increase of 5 even if the projected value is only an increase of 3 |
| How the bug effects the game | Makes the game easier. |
| Issue | The increase in the code was for a +5 not the real +3, changed it to fit projected change. |
| Bugged code | Text  Description automatically generated |
| Reworked code | Text  Description automatically generated |
| Does the change work | Yes |

### Bug 3

|  |  |
| --- | --- |
| What is the bug | The upgrade for ammo kit is showing an increase of 3 even if the projected value is only an increase of 7. |
| How the bug effects the game | Makes the game harder. |
| Issue | The increase in the code was for a +5 not the real +3, changed it to fit projected change. |
| Bugged code | Text  Description automatically generated |
| Reworked code | Text  Description automatically generated |
| Does the change work | Yes |

### Bug 4

|  |  |
| --- | --- |
| What is the bug | If you survive the boss attack and stay alive the game doesn't continue. |
| How the bug effects the game | Makes the game unplayable. |
| Issue | Because I did not add the change for the singleton variable of a boss being present if it touches the tank. |
| Bugged code | A screenshot of a computer  Description automatically generated with medium confidence |
| Reworked code | Text  Description automatically generated |
| Does the change work | Yes |

### Bug 5

|  |  |
| --- | --- |
| What is the bug | If you hit the boss with the large shell when it spawns it will cause it to glitch through world border, trapping them so the game cannot end without quitting |
| How the bug effects the game | Makes the game unplayable. |
| Issue | Because I did not foresee someone breaking the collision box. |
| Bug | Graphical user interface, application, website  Description automatically generated |
| How I will fix | Make a detector that will finds if it out of frame and see if it is out of bounds, if it is then it will teleport it back in frame |
| Reworked code |  |
| Reworked node tree |  |
| Does the change work | Yes |

## Success criteria check

All the criteria have been met, so

|  |  |  |  |
| --- | --- | --- | --- |
| Success Criteria | Fulfilled or not | Where | Why |
| People can make up to 10 accounts on each game instance, so people do not make an infinite amount, filling the storage until there is a computer crash. | Fulfilled | [Registration v1](#_Registration_v1) | The scene checks on the number of accounts and blocks account creation if there is more than ten, this is tested in [prototype 1 Day 09/05/2021 white box testing](#_White_box_testing). |
| The player can exit the game from the HUD. | Fulfilled | [HUD v1](#_HUD_v1) | The scene allows the player to quit the game loop, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player can pause the gameplay. | Fulfilled | [HUD v2](#_White_box_testing_2) | The scene allows the player to pause the game loop, this is seen on [prototype 1 Day 14/05/2021 white box testing.](#_White_box_testing_3) |
| Make sure login works. | Fulfilled | [Login v2](#_Login_v2) | The scene allows the user to login. This is seen on [prototype 1 Day 10/05/2021 white box testing.](#_Login_v2) |
| Make sure the password system is secure | Fulfilled | [Registration v5](#_Registration_v5) | The scene make the person make a secure password, scene on [prototype 3 Day 24/07/2021 white box testing.](#_White_box_testing_4) |
| The person can change input settings (key bindings). | Fulfilled | [Settings v2](#_Settings_v2) | The scene allows the player to change keybindings, this is seen on [prototype 1 Day 13/05/2021 white box testing.](#_White_box_testing_2) |
| The person can change volume settings. | Fulfilled | [Settings v1](#_Settings_v1) | The scene allows the player to volume settings, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The person can change delete all records of their scores. | Fulfilled | [Delete v1](#_Delete_v1) | The scene allows the player to delete their account, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The person can change the sensitivity of movement. | Fulfilled | [Settings v1](#_Settings_v1) | The scene allows the player to sensitivity settings, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The person can change reset all progress. | Fulfilled | [Settings v1](#_Settings_v1) | The scene allows the player to reset their data, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The person can make it full screen. | Fulfilled | [Settings v5](#_Settings_v5) | The scene allows the player to toggle fullscreen, this is seen on [prototype 3 Day 14/07/2021 white box testing.](#_White_box_testing_1) |
| The person can change if they want the HP to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) | The scene allows the player to HUD settings, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The person can change if they want the ammo to be either a bar or number. | Fulfilled | [Settings v1](#_Settings_v1) | The scene allows the player to HUD settings, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The person can change and delete their account. | Fulfilled | [Delete v1](#_Delete_v1)  [Settings v1](#_Settings_v1) | The scenes allows the player to change settings and delete their account, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player can see the amount of in-game cash they have in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) | The scene allows the player to see the amount of cash they got in a game, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player can see the total in-game cash they have on the menu. | Fulfilled | [Menu v1](#_Menu_v1) | This scene shows the current cash player has. Seen on [prototype 1 Day 11/05/2021 white box testing](#_White_box_testing_7) |
| The player can be able to see their top 5 scores. | Fulfilled | [Scores v2](#_Scores_v2) | The scene allows the player to see their scores, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player can be able to see the top 5 scores from all accounts. | Fulfilled | [Scores v2](#_Scores_v2) | The scene allows the player to see the scores of the others, in order, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player can upgrade themselves in the in-game shop with in-game cash. | Fulfilled | [Shop v1](#_Shop_v1) | This scene shows the player to upgrade your character. Seen on [prototype 1 Day 11/05/2021 white box testing](#_White_box_testing_7) |
| Themed music must play in certain scenes. | Fulfilled | [Main v6](#_Main_v6)  [HUD v7](#_HUD_v7) | The scene controls the music. |
| Some lore that is interesting | Fulfilled | [Lore v2](#_Lore_v2) | The scene shows the video lore, seen on [prototype 3 Day 24/07/21 white box testing](#_White_box_testing_4) and <https://youtu.be/3R96poKCVDA?t=225> |
| The enemy moves towards the player and face the player. | Fulfilled | [Zombie v1](#_Zombie_v1) | The scene shows the enemies moving towards the player, seen on [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| The player can rotate | Fulfilled | [Player v1](#_Player_v1) | The scene shows the player rotating, seen on [prototype 1 Day 13/05/2021 white box testing.](#_White_box_testing_2) |
| The shell travels in the direction fired | Fulfilled | [Shell v1](#_Shell_v1) | The scenes shows the a shell traveling in the direction fired, seen on [prototype 1 Day 13/05/2021 white box testing](#_White_box_testing_2) |
| The hook travels in the direction fired | Fulfilled | [Hook v1](#_Hook_v1) | The scenes shows the a hook traveling, seen on [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |
| The kits will appear randomly. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) | The scenes show the ammo and med kits. The testing it is here [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |
| Upon gameplay, the rotation speed is affected by the sensitivity set in the settings. | Fulfilled | [Player v1](#_Player_v1) | The scene shows the player rotating at different speeds due to change in sensitivity |
| The player can fire a tank shell upon the assigned key binding. | Fulfilled | [Shell v1](#_Shell_v1)  [Player v1](#_Player_v1) | The scenes shows the player firing a shell, seen on [prototype 1 Day 13/05/2021 white box testing](#_White_box_testing_2) |
| The player can fire a hook upon assigned key binding. | Fulfilled | [Player v1](#_Player_v1)  [Hook v1](#_Hook_v1) | The scenes shows the player firing a hook, seen on [prototype 1 Day 13/05/2021 white box testing](#_White_box_testing_2) and [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |
| The player’s health must decrease if the enemy touches them and then the enemy disappears. | Fulfilled | [HUD v1](#_HUD_v1) | The scene allows the player to see if they lost health due to an enemy touching them, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The enemy must disappear if touching shell/explosion and increase score and in-game cash. | Fulfilled | [Zombie v1](#_Zombie_v1) | The scene shows the an increase in the score and cash when the enemy dies, seen on [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| The enemies speed and damage done to the player must increase linearly during one gameplay and reset after it. | Fulfilled | [Zombie v1](#_Zombie_v1) | The scene shows the enemies increasing in speed as the game continues, seen on [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| The enemy must appear randomly around the player. | Fulfilled | [Zombie v1](#_Zombie_v1) | The scene shows the enemies randomly appearing, seen on [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| Tank-repair-kits and ammo kits must appear randomly on the screen. | Fulfilled | [Game v4](#_Game_v4)  [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) | The scenes show the ammo and med kits. The testing it is here [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |
| Tank-repair-kits and ammo kits must disappear when touching the hook and increase the appropriate bar. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1)  [Hook v1](#_Hook_v1) | The scenes show the ammo and med kits. When they touches the hook. The testing it is here [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |
| Tank-repair-kits and ammo kits must disappear in a set amount of time if it is not used. | Fulfilled | [AmmoKit v1](#_AmmoKit_v1)  [MedKit v1](#_MedKit_v1) | The scenes show the ammo and med kits disappearing. The testing it is here [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |
| The player is defeated when health is 0. | Fulfilled | [HUD v1](#_HUD_v1)  [Zombie v1](#_Zombie_v1) | The scenes tells the player when their health is 0 and decreases their health, this is seen on [prototype 1 Day 12/05/2021 white box testing](#_White_box_testing_1) and [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| The player can see enemies. | Fulfilled | [Zombie v1](#_Zombie_v1) | The scene shows the enemies, seen on [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| The player can see the playable character. | Fulfilled | [Player v1](#_Player_v1) | The scene shows the player, seen on [prototype 1 Day 13/05/2021 white box testing.](#_White_box_testing_2) |
| The player can interact with the game. | Fulfilled | [Player v1](#_Player_v1) | The scene shows the user interacting with player, seen on [prototype 1 Day 13/05/2021 white box testing.](#_White_box_testing_2) |
| Player can change ammo type | Fulfilled | [HUD v6](#_HUD_v6) | The scene shows the player changing ammo type, seen on [prototype 4 Day 14/07/2021 white box testing.](#_White_box_testing_6) |
| Must have at least 2 variants of enemies, normal and boss | Fulfilled | [Game v6](#_Game_v6)  [Zombie v1](#_Zombie_v1)  [BossZombie v1](#_BossZombie_v1) | The scene loads the boss zombie and zombie, so the game has 2 variants of mobs. Seen on [prototype 3 Day 24/07/21 white box testing](#_White_box_testing_4) and [prototype 1 Day 14/05/2021 white box testing](#_White_box_testing_3) |
| The player can see their ammo and HP levels. | Fulfilled | [HUD v1](#_HUD_v1) | The scene allows the player to see their hp an ammo levels, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player can see the score they got in one gameplay. | Fulfilled | [HUD v1](#_HUD_v1) | The scene allows the player to see their score, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| Must be able to pause and quit the game. | Fulfilled | [HUD v1](#_HUD_v1) | The scene allows the player to pause the game, this is seen on [prototype 1 Day 12/05/2021 white box testing.](#_White_box_testing_1) |
| The player must be able see the amount of cash they got | Fulfilled | [HUD v3](#_Hud_v3) | The scenes show HUD showing they got their cash. The testing it is here [prototype 1 Day 15/05/2021 white box testing](#_White_box_testing_5) |

## Stakeholder evaluation

### Zach Tracy

His evaluation is [here](#_Usability).

### Harry Anthony

Harry believes that it would be ideal to increase the speed of the boss zombie after the player has run out of ammo, as waiting for the game to end is tedious yet also necessary in order to acquire cash for upgrades at the end of the round. Harry also believed that it would be beneficial if the shop opened immediately after a game is over as there are more times when you are available to buy an upgrade than not, creating easier access for new players inexperienced with the games systems.

### Alyvia Mur

Alyvia found the game enjoyable and rather challenging at times, she felt that the game to be better by giving the player a way to quickly end the game if they had run out of ammo on a boss fight, she suggested making the boss speed up and eliminate the tank or even instantly kills so the player doesn’t have to wait long or quit and lose progress. She also felt the game was much easier to play with mouse controls and felt these should be the default settings. Otherwise she found it fun and liked how the difficulty increased overtime.

## Further development

|  |  |
| --- | --- |
| What | Why |
| A skippable tutorial, which explains the controls and how the enemies and ammo system works. | My stakeholder when testing told me that the game experience would be better if he had a skippable tutorial, so he understands how the game works and how to control the player. |
| The last frame in the lore video could be improved. | The stakeholder Zach did not find it interesting. |
| Add more enemy types. | Will make the game more interesting as there is more types. |
| Have a global leaderboard. | It would show the highscores of everyone and it will foster competitiveness making people play more to get better. But to do that a server must be made to keep the scores; another way is to make blockchain and make individual clients the server. |
| Have a way to play as a guest. | People can then play the game without logging in to try it out. |
| Change it so you save progress if you quit game during a boss battle and if you have no ammo. | In a boss round your life depends on the amount of ammo you have, if you run out then it is a slow death until the boss touches the player, so a button to die in a boss round once the ammo is empty will make the game run quicker and be more enjoyable. |
| Shop opens straight after a game | Allows ease of access |

## Limitations

The main limitation now is file size and the hardware limitations of the engine. The hardware limitations could be combated by using a smaller engine like pygame. The file size can be lowered by having lower quality sprites and the use of smaller music tracks. The limitations of the player needing to have hands to play can also be mitigated with a brain computer interface.

## Transcript

### Main.tscn

|  |
| --- |
| extends Control  const Login = preload("res://Scenes/Login.tscn")  const Registration = preload("res://Scenes/Registration.tscn")  const Menu = preload("res://Scenes/Menu.tscn")  const Shop = preload("res://Scenes/Shop.tscn")  const Scores = preload("res://Scenes/Scores.tscn")  const Settings = preload("res://Scenes/Settings.tscn")  const Delete = preload("res://Scenes/Delete.tscn")  const Game = preload("res://Scenes/Game.tscn")  const Controls = preload("res://Scenes/Controls.tscn")  const Credits = preload("res://Scenes/Credits.tscn")  const Lore = preload("res://Scenes/Lore.tscn")  var rng = RandomNumberGenerator.new()  #these are the file locations of scenes that are instanced my main  func \_ready():      Global.rege = 0 #resets which scene the main has loads      backdrop()#loads background      OS.set\_window\_title("Zombie Shooter By Arman")      #changes title of window to it      $Background.show()      #makes the background show      get\_tree().paused = false      #makes sure the game is not paused      Global.cleanup()      #cleans up the userData file      if Global.logged == false:          self.add\_child(Login.instance())          Global.logopen = true      #if someones is not logged in it opens the login scene      var fileCheck = File.new()      var doFileExists = fileCheck.file\_exists("res://HashPass.dat")      var remake = []      if doFileExists == false:          for i in 34:              remake.append("")          remake[33] = "end"          var file = File.new()          file.open("res://HashPass.dat", file.WRITE)          for g in len(remake):              file.store\_line(remake[g])          file.close()      #checks if the hashPass file exists and if it doesn't it makes one  func backdrop():      rng.randomize()      $Background.frame = rng.randi\_range(0,5)      $Objects/RightTank.frame = rng.randi\_range(0,8)      $Objects/LeftTank.frame = rng.randi\_range(0,8)      $Objects/RightTank/Bullet.frame = rng.randi\_range(0,5)      $Objects/LeftTank/Bullet.frame = rng.randi\_range(0,5)      #randomises the background for variety  func \_on\_Music\_finished():      if Global.game == 0:          $Music.play()      else:          $Music.stop()      #if menu music is finished it loops      #it if the game loop is not loaded  func \_process(\_delta):      if Global.logged == true:          AudioServer.set\_bus\_volume\_db(AudioServer.get\_bus\_index("Master"), Global.userInfo.getVolume())      #changes the volume      if Global.rege != 7:          $Objects.show()          #if the game scene is          #not loaded the objects          #are shown          Input.set\_mouse\_mode(Input.MOUSE\_MODE\_VISIBLE)          #makes mouse visible      var flag = false      if Global.logged == true:          if str(Global.userInfo.getVolume()) != "-1.#INF":              if Global.rege != 0:                  if Global.rege != 1:                      if Global.rege != 7:                          if Global.rege != 8:                              if Global.rege != 10:                                  $Music.stream\_paused = false                                  flag = true                                  if $Music.playing == false:                                      $Music.play()      #makes sure mane music plays in scene that have it      if flag == false:          $Music.stream\_paused = true      #if music is not to play it pauses it      if Global.logopen == false:          Global.logopen = true          backdrop()          if Global.rege == 0:              self.add\_child(Login.instance())              $Music.stop()              if OS.window\_fullscreen == true:                  OS.window\_fullscreen = !OS.window\_fullscreen          #if on login is forces it to not be in fullscreen          elif Global.rege == 1:              self.add\_child(Registration.instance())              $Music.stop()          elif Global.rege == 2:              self.add\_child(Menu.instance())          elif Global.rege == 3:              self.add\_child(Shop.instance())          elif Global.rege == 4:              self.add\_child(Scores.instance())          elif Global.rege == 5:              self.add\_child(Settings.instance())          elif Global.rege == 6:              self.add\_child(Delete.instance())          elif Global.rege == 7:              self.add\_child(Game.instance())              backdrop()              $Objects.hide()              get\_tree().paused = false              #hides objects for game loop          elif Global.rege == 8:              self.add\_child(Controls.instance())          elif Global.rege == 9:              self.add\_child(Credits.instance())          elif Global.rege == 10:              self.add\_child(Lore.instance())      #instances the scenes when needed      if Global.button == true:          Global.button = false          if Global.rege != 0:              if Global.rege != 1:                  $SFXButton.play()      #plays button sfx when buttons are pressed      pass  func \_input(event):      if Global.logged == true:          if Global.rege !=6:              if event.is\_action\_pressed("ui\_m"):                  OS.window\_fullscreen = !OS.window\_fullscreen      #if logged in and m is pressed it toggles fullscreen |
| Nodes |
| Display |

### Login.tscn

|  |
| --- |
| extends Control  func \_ready():      $Incorrect.hide()      $LoggedIn.hide()      $Userbox.editable = true      $Passbox.editable = true      $Log.disabled = false      $Quit.disabled = false      $Reg.disabled = false      $Passbox.text = ""      $Userbox.text = ""      pass      #resets login  func \_on\_Log\_pressed():      Global.button = true      $Userbox.editable = false      $Passbox.editable = false      $Log.disabled = true      $Quit.disabled = true      $Reg.disabled = true      #disable boxes to make sure no errors occur      var user = $Userbox.text      var find = searchUser(user)      #checks to see if such a username exists      if find == false:          $Incorrect.show()          yield(get\_tree().create\_timer(2.0), "timeout")          \_ready()          return      #if username is not found it shows there is an error      var passw = $Passbox.text      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      #hashes password and finds the index      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      file.close()      #copy and splits by line the passwords      var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          if content[num] == hashed:              Global.logged = true              break          elif content[num] == "":              break          elif content[num] == " ":              break          elif content[num] == "end":              num = 0          if count > 10:              return false          num = num +1      #loops to find the password      if Global.logged == true:          $LoggedIn.show()          yield(get\_tree().create\_timer(0.5), "timeout")          Global.rege = 2          Global.logopen = false          self.queue\_free()      #tells the program the user was found      else:          $Incorrect.show()          yield(get\_tree().create\_timer(2.0), "timeout")          \_ready()      #tells program that user wasn't found so resets and      #tells user incorrect passwords  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      #copys the file and splits by line      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == "":              return false          if  userline[0] == " ":              return false          if  userline[0] == user:              Global.userInfo = Global.user.new(userline)              return true      return true      #tries to find if the user exists in database  func \_on\_Reg\_pressed():      Global.button = true      Global.rege = 1      Global.logopen = false      self.queue\_free()      #makes the button make button noise      #makes main load registration scene      #self terminates this scene  func \_on\_Quit\_pressed():      get\_tree().quit()      #closes whole program  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Log\_pressed()      #if enter is pressed      #it tries to login |
| Nodes |
| Scene |

### Registration.tscn

|  |
| --- |
| extends Control  var alphabet = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"]  var types = [false,false,false,false,true]  #getting the flags ready to check password with  func \_ready():      $Max.hide()      $Back.disabled = false      $Quit.disabled = false      $Reg.disabled = false      $Userbox.editable = true      $Passbox.editable = true      $ConPassbox.editable = true      #to reset nodes when instanced or failed      #registration      Global.cleanup()      var file = File.new()      var count = 0      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var arrcontent = content.split("\n")      for n in len(arrcontent):          count += 1      #counts the number of users on current      #software      if count >= 11:          $Max.show()          $Reg.hide()          $Userbox.hide()          $Passbox.hide()          $ConPassbox.hide()      #if it is >=11 than 11 it disables      #registartion      pass  func searchUser(user):      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              wrong(1)              return false          if  userline[0] == "":              return true          if  userline[0] == " ":              return true      return true      #checks if that username already exists  func letterCheck(list):      var flag = false      for i in range(0, 26):          if list.has(alphabet[i]) == true:              flag = true              break      types[1] = flag      #checks if password has normal      #characters    func symbolCheck(list):      var listSymbols = ["`","¬","!","£","$","%","^","&","\*","(",")","\_","-","+","=","{","[","}","]",";","@","~","#","<",",",">",".","?","/",":","|"]      var flag = false      for i in len(listSymbols)-1:          if list.has(listSymbols[i].to\_upper()) == true:              flag = true              break      types[3] = flag      #checks to see if password has symbols  func capLetterCheck(list):      var flag = false      for i in len(alphabet)-1:          if list.has(alphabet[i].to\_upper()) == true:              flag = true              break      types[2] = flag      #checks to see if password has capital characters  func number(list):      var flag = false      for i in range(0,10):          if list.has(str(i)) == true:              flag = true              break      types[0] = flag      #checks to see if password has      #a number  func passCheck(passw):      types[0] = false      types[1] = false      types[2] = false      types[3] = false      types[4] = true      #resets flags for password      var list =[]      for g in len(passw):          list.append(str(passw[g]))          if passw[g] == "'":              wrong(2)              return false          elif passw[g] == '"':              wrong(2)              return false          elif passw[g] == ',':              wrong(2)              return false      #checks password if it has      #'", and tells that they are illegal      number(list)      letterCheck(list)      capLetterCheck(list)      symbolCheck(list)      #does all checks      if len(list) <= 5:          types[4] = false      #checks if password is longer      #than 5 characters      for i in len(types):          if types[i] == false:              wrong(5)              return false      return true      #checks to see if all flags are      #true  func addLogin(user, passw):      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      #gets all data on userdata      var file2 = File.new()      file2.open("res://userData.dat", file.WRITE)      var add = content+"\n"+user +",0,20,10,2,10,1,0.501187,0,0,0,0,0,3,1,1,0,1"      file2.store\_line(add)      file2.close()      #adds user to userdata with basic info      var loco = Global.hashedLocation(user)      #finds location to store password      var hashed = passw.sha256\_text()      #gets hashed password      var file3 = File.new()      file3.open("res://HashPass.dat", file.READ)      var content2 = file3.get\_as\_text()      content2 = content2.split("\n", true)      file3.close()      #gets all data on HashPass      var file4 = File.new()      file4.open("res://HashPass.dat", file.WRITE)      content2[loco] = hashed      for i in len(content2)-1:          file4.store\_line(str(content2[i]))      file4.close()      #writes update to HashPass to include      #new password of new user  func wrong(type):      if type == 1:          $Wrong.set\_text("Username already in use")          $Wrong.popup()      elif type == 2:          var text = 'Password is illegal, cannot have the characters " or ' + "'"          $Wrong.set\_text(text)          $Wrong.popup()      elif type == 3:          $Wrong.set\_text("Passwords do not match")          $Wrong.popup()      elif type == 4:          var text = 'Username is illegal, cannot have the characters " or' + "' or , "          $Wrong.set\_text(text)          $Wrong.popup()      elif type == 5:          var text = "Password is not complex enough, must have at least: "          if types[0] == false:              text = text + "a number, "          if types[1] == false:              text = text + "a letter, "          if types[2] == false:              text = text + "a capital letter, "          if types[3] == false:              text = text + "a symbol, "          if types[4] == false:              text = text + "at least 6 characters long, "          $Wrong.set\_text(text)          $Wrong.popup()      #opens dialog and tells the error of current registration data  func \_on\_Wrong\_popup\_hide():      \_on\_Wrong\_confirmed()      #if you click out of the dialog      #it counts as a confirmed  func \_on\_Wrong\_confirmed():      yield(get\_tree().create\_timer(0.5), "timeout")      \_ready()      #resets data when input for      #registration fails  func userCheck(userName):      var list = []      for g in len(userName):          list.append(str(userName[g]))      for i in len(list)-1:          if ord(list[i]) == ord("'"):              wrong(4)              return false          if ord(list[i]) == ord('"'):              wrong(4)              return false          if ord(list[i]) == ord(","):              wrong(4)              return false      #checks if username has any illegal      #characters      if len(list) < 3:          wrong(4)          return false      else:          return true      #checks if username      #is longer than 3      #letters  func \_on\_Reg\_pressed():      $Back.disabled = true      $Quit.disabled = true      $Reg.disabled = true      $Userbox.editable = false      $Passbox.editable = false      $ConPassbox.editable = false      #blocks all editable nodes      Global.button = true      #makes the button sfx to play      var user = $Userbox.text      var goodUser = userCheck(user)      #gets username from node and      #checks if it valid      var passw = $Passbox.text      #gets password from node      if goodUser == true:          var findu = searchUser(user)          if $Passbox.text == $ConPassbox.text:              #makes sure passwords match between              #password node and confirmed password              #node              if findu == false:                  pass              #sees if username if already there              if findu == true:                  var findp = passCheck(passw)                  if findp == false:                      pass                  #checks if password is valid                  if findp == true:                      addLogin(user,passw)                      Global.cleanup()                      Global.rege = 0                      Global.logopen = false                      self.queue\_free()                      #adds new user and causes                      #login scene to load          else:              wrong(3)              pass              #if passwords don't match it makes              #dialog load  func \_on\_Back\_pressed():      Global.button = true      Global.rege = 0      Global.logopen = false      self.queue\_free()      #if they want to go back      #to login this runs      #make button sfx load      #and loads login scene  func \_on\_Quit\_pressed():      Global.button = true      get\_tree().quit()      #closes app if quit is pressed |
| Nodes |
| Scene |

### Menu.tscn

|  |
| --- |
| extends Control  var type = -1  #makes type invalid so if it loads now nothing happens  func \_ready():      $Money.text = "Total Cash:" + str(Global.userInfo.getCash())      #shows total cash of user  func \_on\_Logout\_pressed():      Global.button = true      type = 1      stop()      #plays button sfx      #starts dialog box  func \_on\_Quit\_pressed():      Global.button = true      type = 0      stop()      #plays button sfx      #starts dialog box  func stop():      if type == 0:          $Change.set\_text("Press ok to confirm you are closing the game.")          $Change.popup()      if type == 1:          $Change.set\_text("Press ok to confirm you are logging out.")          $Change.popup()      #loads dialog for the scenarios  func \_on\_Change\_confirmed():      Global.button = true      if type == 0:          get\_tree().quit()      #if dialog is confirmed for type 0 it quits app      if type == 1:          Global.logged = false          Global.rege = 0          Global.logopen = false          self.queue\_free()      #if dialog is confirmed for type 1 it logout user  func \_on\_Shop\_pressed():      Global.button = true      Global.rege = 3      Global.logopen = false      self.queue\_free()      #loads shop scene  func \_on\_Game\_pressed():      Global.button = true      Global.rege = 7      Global.logopen = false      self.queue\_free()      #loads game scene  func \_on\_Settings\_pressed():      Global.button = true      Global.rege = 5      Global.logopen = false      self.queue\_free()      #loads settings scene  func \_on\_Controls\_pressed():      Global.button = true      Global.rege = 8      Global.logopen = false      self.queue\_free()      #loads controls scene    func \_on\_Credits\_pressed():      Global.button = true      Global.rege = 9      Global.logopen = false      self.queue\_free()      #loads credits scene  func \_on\_Scores\_pressed():      Global.button = true      Global.rege = 4      Global.logopen = false      self.queue\_free()      #loads scores scene  func \_on\_Lore\_pressed():      Global.button = true      Global.rege = 10      Global.logopen = false      self.queue\_free()      #loads lore scene |
| Nodes |
| Scene |

### globals.gd

|  |
| --- |
| extends Node  var logopen = false  #tells if a scene is currently open  var rege = 0  #tells what scene is currently loaded and what scene to load  var logged = false  #tlls if a user is logged in  var pause = 0  #tells if game is paused right now  # if 1 means pause game play, if 0 means let gameplay continue  var game = 0 # 0 means no game, 1 means currently playing  var hp = 1  #tells current hp for player for game loop  var ammo = 1  #tells current ammo for player for game loop  var cash = 0  #tells cash a person got in a single game loop  var score = 0  #tells score of a person got in 1 game loop  var newGame = 0  #tells if user wants to play again  var firing = 0  #tells is user is currently firing  var zombieNum = 0  #tells the num of zombies on screen  export (float) var ticks = 0  #tells the number of in game ticks since  #game loop has started  var hook = 0  #tells if user is currently firing hook  var mKit = 0  #tells if a med kit is loaded  var aKit = 0  #tells if a ammokit is loaded  var button = false  #tells if a button is pressed  var typeShot = 1  #tells what type of ammo is currently loaded  # 1-normal 2-incendiary 3 large-Normal  var userInfo  #object  var boss = false  #tells if a boss if currently loaded  func cleanup():      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      var counter = len(content)-1      for i in len(content):          if content[counter] == "":              content.remove(counter)          elif content[counter] == " ":              content.remove(counter)          counter = counter - 1      file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      #cleans up userdata file to remove gaps  class user:      #the class that used to      #instance a user’s data      var name      var cash      var maxShell      var maxHP      var kitHP      var kitAmmo      var sens      var volume      var h1      var h2      var h3      var h4      var h5      var Hudtype      var rotateType      var fireType      var colour      var hook      #all attributes of a user      func \_init(data):          name = data[0]          cash = data[1]          maxShell = data[2]          maxHP = data[3]          kitHP = data[4]          kitAmmo = data[5]          sens = data[6]          volume = data[7]          h1 = data[8]          h2 = data[9]          h3 = data[10]          h4 = data[11]          h5 = data[12]          Hudtype = data[13]          rotateType =data[14]          fireType = data[15]          colour = data[16]          hook = data[17]      #initalising all attributes of a user      func getName():          return name      func getCash():          return int(cash)      func setCash(new):          cash = new      func getMaxShell():          return int(maxShell)      func setMaxShell(new):          maxShell = new      func getMaxHP():          return int(maxHP)      func setMaxHP(new):          maxHP = new      func getkitHP():          return int(kitHP)      func setkitHP(new):          kitHP = new      func getkitAmmo():          return int(kitAmmo)      func setkitAmmo(new):          kitAmmo = new      func getSens():          return int(sens)      func setSens(new):          sens = new      func getVolume():          return int(volume)      func setVolume(new):          volume = new      func getH1():          return h1      func setH1(new):          h1 = new      func getH2():          return h2      func setH2(new):          h2 = new      func getH3():          return h3      func setH3(new):          h3 = new      func getH4():          return h4      func setH4(new):          h4 = new      func getH5():          return h5      func setH5(new):          h5 = new      func getHud():          return int(Hudtype)      func setHud(new):          Hudtype = new      func getRotate():          return int(rotateType)      func setRotate(new):          rotateType = new      func getFire():          return int(fireType)      func setFire(new):          fireType = new      func getColour():          return int(colour)      func setColour(new):          colour = new      func getHook():          return int(hook)      func setHook(new):          hook = new      #getters and setters for all      #attributes      func getALL():          var string =  str(name)+","+str(cash)+","+str(maxShell)+","+str(maxHP)+","+str(kitHP)+","+str(kitAmmo)+","+str(sens)+","+str(volume)+","+str(h1)+","+str(h2)+","+str(h3)+","+str(h4)+","+str(h5)+","+str(Hudtype)+","+str(rotateType)+","+str(fireType)+","+str(colour)+","+str(hook)          return string      #to get all user's data as  string  func hashedLocation(user):      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      file.close()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          if content[num] == "del":              return num          elif content[num] == "":              return num          elif content[num] == " ":              return num          elif content[num] == "end":              num = 0          elif count > 10:              return false          num = num +1      #gives where a password is stored      #in hash table  func save\_changes():      Global.userInfo.setCash(Global.userInfo.getCash()+Global.cash)      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      var replacing = userInfo.getALL()      file.close()      var counter = len(content)-1      for i in len(content):          var splitContent = content[i].split(",",true)          if splitContent[0] == userInfo.getName():              content[i] = replacing          counter = counter - 1      file.open("res://userData.dat", file.WRITE)      for g in len(content):          file.store\_line(content[g])      file.close()      cleanup()      #saves all changes |

### Shop.tscn

|  |
| --- |
| extends Control  onready var newHP  onready var newHP\_cost  onready var newAmmo  onready var newAmmo\_cost  onready var newMed  onready var newMed\_cost  onready var newAmmok  onready var newAmmok\_cost  #loads all variable for scene  func \_ready():      newHP = Global.userInfo.getMaxHP()+2      newHP\_cost = int(newHP\*2)      newAmmo = Global.userInfo.getMaxShell()+3      newAmmo\_cost = int(newAmmo\*1.5)      newMed = Global.userInfo.getkitHP()+1      newMed\_cost = int(newMed\*2)      newAmmok = Global.userInfo.getkitAmmo()+7      newAmmok\_cost = int(newAmmok\*1.5)      #calculates upgrade costs      $Money.text = "Total Cash:" + str(Global.userInfo.getCash())      var hptext = "Upgrade Max HP Currently:" + str(Global.userInfo.getMaxHP()) +", upgraded:"+ str(newHP)+ ", upgrade cost:" + str(newHP\_cost)      $Hp.text = str(hptext)      var ammotext = "Upgrade Max Ammo, Currently:" + str(Global.userInfo.getMaxShell()) + ", upgraded:"+ str(newAmmo)+ ", upgrade cost:" + str(newAmmo\_cost)      $Ammo.text = str(ammotext)      var med =  "Upgrade Medkit regen amount, Currently:" + str(Global.userInfo.getkitHP()) +", upgraded:"+str(newMed)+",upgrade cost:" + str(newMed\_cost)      $Med.text = str(med)      var kit = "Upgrade Ammokit regen amount, Currently:" + str(Global.userInfo.getkitAmmo()) + ", upgraded:" + str(newAmmok) +",upgrade cost:" + str(newAmmok\_cost)      $Kit.text = str(kit)      #loads text for all buttons  func change(cost):      Global.userInfo.setCash(Global.userInfo.getCash()-cost)      Global.save\_changes()      \_ready()      pass      #saves changes  func check(cost):      if Global.userInfo.getCash() >= cost:          return true      else:          return false      #checks if user has enough cash      #to do purchase  func notEnough():      $NotEnough.show()      yield(get\_tree().create\_timer(1.0), "timeout")      $NotEnough.hide()      pass      #loads node if the user has not enough cash  func \_on\_AmmoMax\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newAmmo\_cost)      if enoughCash == true:          Global.userInfo.setMaxShell(Global.userInfo.getMaxShell()+3)          change(newAmmo\_cost)      else:          notEnough()      pass      #changes max ammo of user  func \_on\_Ammokit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newAmmok\_cost)      if enoughCash == true:          Global.userInfo.setkitAmmo(Global.userInfo.getkitAmmo()+7)          change(newAmmok\_cost)      else:          notEnough()      pass      #changes ammo kit size of user  func \_on\_HP\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newHP\_cost)      if enoughCash == true:          Global.userInfo.setMaxHP(Global.userInfo.getMaxHP()+2)          change(newHP\_cost)      else:          notEnough()      pass      #changes max hp of user  func \_on\_MedKit\_pressed():      Global.button = true      yield(get\_tree().create\_timer(0.01), "timeout")      var enoughCash = check(newMed\_cost)      if enoughCash == true:          Global.userInfo.setkitHP(Global.userInfo.getkitHP()+1)          change(newMed\_cost)      else:          notEnough()      pass      #changes size of medkit  func \_on\_Menu\_pressed():      Global.button = true      Global.rege = 2      Global.logopen = false      self.queue\_free()      #makes menu load  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Menu\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Menu\_pressed()      #if enter or esc pressed it loads menu |
| Nodes |
| Scene |

### Scores.tscn

|  |
| --- |
| extends Control  func \_ready():      var allScores =[]      $PersonalScores.text = "1. "+Global.userInfo.getH1()+"\n"+"2. "+Global.userInfo.getH2()+"\n"+"3. "+Global.userInfo.getH3()+"\n"+"4. "+Global.userInfo.getH4()+"\n"+"5. "+Global.userInfo.getH5()+"\n"      #loads all personal scores into list      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      content = content.split("\n",true)      file.close()      #gets all user data      for i in len(content)-1:          var line = content[i]          line = line.split(",", true)          var add1 = line[0]+". "+line[8]          var add2 = line[0]+". "+line[9]          var add3 = line[0]+". "+line[10]          var add4 = line[0]+". "+line[11]          var add5 = line[0]+". "+line[12]          allScores.append(add1)          allScores.append(add2)          allScores.append(add3)          allScores.append(add4)          allScores.append(add5)      #splits them to find username and scores      for i in range(allScores.size()-1, -1, -1):          for j in range(1,i+1,1):              var notes = allScores[j].split(". ", true)              var prevnotes = allScores[j-1].split(". ", true)              if int(notes[1]) > int(prevnotes[1]):                  var temp = allScores[j-1]                  allScores[j-1] = allScores[j]                  allScores[j] = temp      #sorts them by number, highest first      var scoresOrdered = ""      for i in len(allScores):          scoresOrdered = scoresOrdered + allScores[i] + "\n"      $HighScores.text = scoresOrdered      #loads score into node  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Back\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Back\_pressed()      #goes back to menu when enter or esc pressed    func \_on\_Back\_pressed():      Global.button = true      Global.rege = 2      Global.logopen = false      self.queue\_free() |
| Nodes |
| Scene |

### Controls.tscn

|  |
| --- |
| extends Control  func \_ready():      var movement = ""      var shootS = ""      var hook = ""      if Global.userInfo.getRotate() == 1:          movement = "When the Left and Right Arrow Keys are pressed."          if str(Global.userInfo.getVolume()) != "-1.#INF":              $Lr.play()      elif Global.userInfo.getRotate() == 2:          movement = "When the A and D keys are pressed."          if str(Global.userInfo.getVolume()) != "-1.#INF":              $Ad.play()      else:          movement = "Using the cursur location."          if str(Global.userInfo.getVolume()) != "-1.#INF":              $Mouse.play()      #plays the speech version of what is shown in controls for movement      if Global.userInfo.getFire() == 3:          shootS = "When the R key is pressed."      elif Global.userInfo.getFire() == 2:          shootS = "When the Space bar is pressed."      elif Global.userInfo.getFire() == 1 :          shootS = "When the F key is pressed."      else:          shootS = "When the left mouse button is pressed"      #loads text for how user fires shells      if Global.userInfo.getHook() == 1:          hook = "When H key is pressed."      elif Global.userInfo.getHook() == 2:          hook = "When the Shift key is pressed."      else:          hook = "Using the right mouse button."      #loads text on how user fires hook      var text = " Movement of player is: "+ movement      text = text + "\n How to shoot the Shell: "+ shootS      text = text + "\n How fire the hook: " + hook      text = text + "\n Press 1 in game for Normal Shell, takes 1 ammo."      text = text + "\n Press 2 in game for Incendiary shell, takes 2 ammo."      text = text + "\n Press 3 in game for Large normal shell, takes 3 ammo."      text = text + "\n To toggle full screen press the m key."      $Stuff.text = text      #loads rest of info and writes the node  func \_Rotate\_Finished():      if Global.userInfo.getFire() == 3:          $Rk.play()      elif Global.userInfo.getFire() == 2:          $Space.play()      elif Global.userInfo.getFire() == 1 :          $Fk.play()      else:          $ClickL.play()      #plays on how user fires shells  func \_Fire\_Finished():      if Global.userInfo.getHook() == 1:          $Hk.play()      elif Global.userInfo.getHook() == 2:          $Shift.play()      else:          $ClickR.play()      #plays on how user fires hook  func \_Hook\_Finished():      $Shell.play()      #plays on types of shells  func \_on\_Shell\_finished():      $Screen.play()      #plays on how to toggle      #fullscreen    func \_on\_Menu\_pressed():      Global.button = true      Global.rege = 2      Global.logopen = false      self.queue\_free()      #loads menu  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Menu\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Menu\_pressed()          #loads menu |
| Nodes |
| Scene |

### Credits.tscn

|  |
| --- |
| extends Control  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      Global.button = true      self.queue\_free()      #loads menu    func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Back\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Back\_pressed()      #loads menu |
| Nodes |
| Scene |

### Lore.tscn

|  |
| --- |
| extends Control  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Back\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Back\_pressed()      #loads menu  func \_ready():      $Replay.hide()      $Skip.show()      $Fallen.hide()      $Finished.hide()      $VideoPlayer.show()      $VideoPlayer.play()      #resets videoplayer  func \_on\_VideoPlayer\_finished():      $VideoPlayer.hide()      $Skip.hide()      $Finished.show()      $Fallen.show()      $Replay.show()      #hides videoplayer      #loads img  func \_on\_Replay\_pressed():      \_ready()      #replays video  func \_on\_Replay\_mouse\_entered():      $Replay.modulate = Color(0,0,0)    func \_on\_Replay\_mouse\_exited():      $Replay.modulate = Color(1,1,1)  func \_on\_Back\_pressed():      Global.rege = 2      Global.logopen = false      Global.button = true      self.queue\_free()      #loads menu  func \_on\_Skip\_pressed():      $VideoPlayer.stop()      \_on\_VideoPlayer\_finished()      #skips video to img  func \_on\_Back\_mouse\_entered():      $Back.modulate = Color(0,0,0)  func \_on\_Back\_mouse\_exited():      $Back.modulate = Color(1,1,1)  func \_on\_Skip\_mouse\_entered():      $Skip.modulate = Color(0,0,0)  func \_on\_Skip\_mouse\_exited():      $Skip.modulate = Color(1,1,1) |
| Nodes |
| Scene |

### Settings.tscn

|  |
| --- |
| extends Control  func \_ready():      $Sens.value = Global.userInfo.getSens()      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userInfo.getSens())      #gets and sets current movement sensitivity      if Global.userInfo.getRotate() == 3:          $Sens.hide()      else:          $Sens.show()      #hides sensitivity editor if using mouse to move      var num = db2linear(Global.userInfo.getVolume())      $Vol.value = num      $Vol/Label.text = "Volume is currently:" + str(int(num\*100))      #gets and sets current volume      $HUD.add\_item("Change on how you see the HUD in Game")      $HUD.set\_item\_disabled(0, true)      $HUD.add\_item("Have only the numbers")      $HUD.add\_item("Have only the bar")      $HUD.add\_item("Have both")      #sets all options to HUD type      $Movement.add\_item("Change on how you rotate player")      $Movement.set\_item\_disabled(0, true)      $Movement.add\_item("Left and Right arrow keys")      $Movement.add\_item("Using the A and D keys")      $Movement.add\_item("Using the Mouse location")      #sets all options for movement      $Fire.add\_item("Change on how you fire")      $Fire.set\_item\_disabled(0, true)      $Fire.add\_item("The F key")      $Fire.add\_item("The Space bar")      $Fire.add\_item("The R key")      $Fire.add\_item("Using a left click")      #sets all options to fire shells      $Colour.add\_item("Change Tank colour")      $Colour.set\_item\_disabled(0, true)      $Colour.add\_item("Black")      $Colour.add\_item("Grey")      $Colour.add\_item("White")      $Colour.add\_item("Purple")      $Colour.add\_item("Brown")      $Colour.add\_item("Orange")      $Colour.add\_item("Red")      $Colour.add\_item("Green")      $Colour.add\_item("Blue")      #sets all options to change tank colour      $Hook.add\_item("Change on how you fire hook")      $Hook.set\_item\_disabled(0, true)      $Hook.add\_item("The H key")      $Hook.add\_item("Shift key")      $Hook.add\_item("Using the Right click")      #sets all options to fire hook  func \_on\_Sens\_value\_changed(value):      value = str(value)      Global.userInfo.setSens(value)      $Sens/Label.text = "Sensitivity of Turret, Currently: " + str(Global.userInfo.getSens())      #sets new sensitivity when it changes  func \_on\_Vol\_value\_changed(num):      $Vol/Label.text = "Volume is currently:" + str(int(num\*100))      num = linear2db(num)      Global.userInfo.setVolume(num)      #sets new volume when it changes  func \_on\_Delete\_pressed():      Global.button = true      Global.rege = 6      Global.logopen = false      self.queue\_free()      #loads delete scene    func \_on\_Reset\_pressed():      $Res.popup()      #loads rest popup  func \_on\_Res\_confirmed():      Global.button = true      Global.userInfo.setCash("0")      Global.userInfo.setMaxShell("20")      Global.userInfo.setMaxHP("10")      Global.userInfo.setkitHP("2")      Global.userInfo.setkitAmmo("10")      Global.userInfo.setSens("1")      Global.userInfo.setVolume("0.501187")      Global.userInfo.setH1("0")      Global.userInfo.setH2("0")      Global.userInfo.setH3("0")      Global.userInfo.setH4("0")      Global.userInfo.setH5("0")      Global.userInfo.setHud("3")      Global.userInfo.setRotate("1")      Global.userInfo.setFire("1")      Global.userInfo.setColour("0")      Global.userInfo.setHook("1")      Global.save\_changes()      \_on\_Menu\_pressed()      #resets al users attributes  func \_on\_HUD\_item\_selected(type):      Global.button = true      Global.userInfo.setHud(type)      #chanegs HUD type  func \_input(event):      if event.is\_action\_pressed("ui\_accept"):          \_on\_Menu\_pressed()      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Menu\_pressed()      #loads menu  func \_on\_Menu\_pressed():      Global.button = true      Global.save\_changes()      Global.rege = 2      Global.logopen = false      self.queue\_free()      #loads menu  func \_on\_Movement\_item\_selected(type):      Global.button = true       # how to rotate 1 = left and right arrow 2 = a d keys 3=mouse location      if type == 3:          $Sens.hide()      else:          $Sens.show()      Global.userInfo.setRotate(type)      #changes control for movement  func \_on\_Fire\_item\_selected(type):      Global.button = true      #how to fire 1 = f 2 =space 3 = r 4=mouse click      Global.userInfo.setFire(type)      #changes control for firing of shells  func \_on\_Fullscreen\_pressed():      OS.window\_fullscreen = !OS.window\_fullscreen      #toggles fullscreen  func \_on\_Colour\_item\_selected(type):      type = type - 1      Global.button = true      Global.userInfo.setColour(type)      #sets new colour for tank  func \_on\_Hook\_item\_selected(index):      #how to fire 1 = h, 3 = right, 2= shift      Global.button = true      Global.userInfo.setHook(index)      #sets on how hook is fired |
| Nodes |
| Scene |

### Delete.tscn

|  |
| --- |
| extends Control  func \_ready():      $Incorrect.hide()      #hide it  func \_on\_Settings\_pressed():      Global.button = true      Global.rege = 5      Global.logopen = false      self.queue\_free()      #loads settings  func searchUser(user): # see if given username matches one in file      var file = File.new()      file.open("res://userData.dat", file.READ)      var content = file.get\_as\_text()      file.close()      var users = content.split("\n",true)      var userline = users[0]      for i in len(users):          userline = users[i]          userline = userline.split(",",true)          if  userline[0] == user:              return true      return false      #checks if such a username exists  func incorrectLogin():      $Incorrect.show()      yield(get\_tree().create\_timer(2.0), "timeout")      $Incorrect.hide()      $Userbox.text = ""      $Passbox.text = ""      #load incorrect  func \_input(event):      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Settings\_pressed()      #loads settings  func \_on\_Delete\_pressed():      $Del.set\_text("Are you sure you want to delete your account")      $Del.popup()      #loads popup for confirmation  func \_on\_Del\_confirmed():      Global.button = true      var userName = $Userbox.text      var passWord = $Passbox.text      var foundUser = searchUser(userName)      if foundUser == false:          incorrectLogin()          pass      #checks if username exists      elif userName != Global.userInfo.getName():          incorrectLogin()          pass      #checks if username matches current users name      else:          var foundPass = realPass(userName, passWord)          if foundPass == -1:              incorrectLogin()              pass          #checks if password is found          else:              var file = File.new()              file.open("res://userData.dat", file.READ)              var content = file.get\_as\_text()              content = content.split("\n",true)              #gets all userData and split by line to make list              file.close()              var counter = len(content)-1              for i in len(content):                  var splitContent = content[i].split(",",true)                  if splitContent[0] == userName:                      content[i] = ""                  counter = counter - 1              #finds index of user in userdata and expunges it              file.open("res://userData.dat", file.WRITE)              for g in len(content):                  file.store\_line(content[g])              file.close()              Global.cleanup()              #saves and cleans userdata              file.open("res://HashPass.dat", file.READ)              content = []              content = file.get\_as\_text().split("\n",true)              file.close()              #gets all data in HashPass and split by line              content[foundPass] = "del"              #deletes current user password              file.open("res://HashPass.dat", file.WRITE)              for i in len(content)-1:                  file.store\_line(str(content[i]))              #saves updated HashPass              file.close()              Global.rege = 0              Global.logopen = false  Global.logged = false              self.queue\_free()              #loads login scene  func realPass(user ,passw):      var hashed = passw.sha256\_text()      var listUser = []      for g in len(user):          listUser.append(user[g])      var num = 0      for i in len(listUser):          num = num + ord(listUser[i])      while num > 32:          num = num /32      var file = File.new()      file.open("res://HashPass.dat", file.READ)      var content = []      content = file.get\_as\_text().split("\n",true)      file.close()      var count = 0      var loop = true      while loop == true:          if num >= 32:              num = 0          count = count + 1          if content[num] == hashed:              return  num          elif content[num] == "":              return -1          elif content[num] == " ":              return -1          elif content[num] == "end":              num = 0          if count > 10:              return -1          num = num +1  #finds where password is stored in hash table |
| Nodes |
| Scene |

### Game.tscn

|  |
| --- |
| extends Node2D  const HUD = preload("res://Scenes/HUD.tscn")  const Player = preload("res://Scenes/Player.tscn")  const Zombie = preload("res://Scenes/Zombie.tscn")  const BossZombie = preload("res://Scenes/BossZombie.tscn")  const AmmoKit = preload("res://Scenes/AmmoKit.tscn")  const MedKit = preload("res://Scenes/MedKit.tscn")  #gets all paths to nodes it will instance  var launchZ = 0  var launchA = 0  var launchM = 0  var num = 1.0  var count =1.0  export (float) var time = 0  #loads variable that are global  func waitz():      yield(get\_tree().create\_timer(1.5), "timeout")      launchZ = 0  #wait between spawns of zombies  func waitm():      yield(get\_tree().create\_timer(20.0), "timeout")      launchM = 0  #wait between spawns of medkits  func waita():      yield(get\_tree().create\_timer(20.0), "timeout")      launchA = 0  #wait between spawns of ammokits  func \_ready():      get\_tree().paused = false      Global.pause = 0      Global.game = 1      Global.newGame = 0      Global.ticks = 0      Global.zombieNum = 0      Global.hook = 0      Global.aKit = 0      Global.mKit = 0      Global.boss = false      Global.typeShot = 1      #resets all game variables      self.add\_child(Player.instance())      self.add\_child(HUD.instance())      #instances HUD and player      var loop = true      while loop == true:          yield(get\_tree().create\_timer(3.0), "timeout")          if Global.pause == 0:              if Global.boss == false:                  Global.ticks = Global.ticks + 0.5      #increment game ticks every 3 seconds by 0.5  func \_process(\_delta):      if Global.game == 0:          Global.rege = 2          Global.logopen = false          self.queue\_free()      #makes game load menu is Global.game == 0      #as that means the game has quit      elif Global.newGame == 1:          Global.logopen = false          self.queue\_free()      #makes itself self-terminate then load again      elif Global.pause == 1:          pass      #make nothing happen as game is paused      else:          if Global.zombieNum < 3:              if Global.boss == false:                  self.add\_child(Zombie.instance())                  Global.zombieNum =Global.zombieNum  +  1              #instances zombies if there are less than 3 on screen          if launchZ == 0:              if Global.zombieNum < 20:                  if Global.boss == false:                      self.add\_child(Zombie.instance())                      Global.zombieNum =Global.zombieNum  +  1                      launchZ = 1                      waitz()                  #instances zombie every 1.5s          if launchM == 0:              if Global.mKit == 0:                  if Global.boss == false:                      if Global.hp < Global.userInfo.getMaxHP():                          self.add\_child(MedKit.instance())                          Global.mKit = 1                          launchM = 1                          waitm()                      #instances medkit every 10s if player has less than max hp          if launchA == 0:              if Global.aKit == 0:                  if Global.boss == false:                      if Global.ammo < Global.userInfo.getMaxShell():                          self.add\_child(AmmoKit.instance())                          Global.aKit = 1                          launchA = 1                          waita()                      #instances ammokit every 10s if payer has less than max ammo          num = Global.ticks/10          if num == count:              if Global.boss == false:                  Global.boss = true                  count = count +1.0                  self.add\_child(BossZombie.instance())                  Global.ticks = Global.ticks + 0.5              #if ticks is a multiple of 10 it loads a boss |
| Node |
| Scene |

### HUD.tscn

|  |
| --- |
| extends Control  var Hudtype = Global.userInfo.getHud()  var scores = []  var getCash = Global.cash  #loads global varaibles  func \_ready():      $ProHUDHP.set\_max(Global.userInfo.getMaxHP())      $ProHUDAmmo.set\_max(Global.userInfo.getMaxShell())      Global.hp = int(Global.userInfo.getMaxHP())      Global.ammo = int(Global.userInfo.getMaxShell())      Global.cash = 0      Global.score = 0      $Unpause.hide()      $Quit.hide()      $Over.hide()      #sets varaibles for nodes and hides pausing and quiting nodes      if str(Global.userInfo.getVolume()) != "-1.#INF":          $Play.play()      #plays game music      scores.append(int(Global.userInfo.getH1()))      scores.append(int(Global.userInfo.getH2()))      scores.append(int(Global.userInfo.getH3()))      scores.append(int(Global.userInfo.getH4()))      scores.append(int(Global.userInfo.getH5()))      #stores current top 5 scores      pass    func \_on\_Pause\_pressed():      Global.button = true      $Pause.hide()      Global.pause = 1      $Play.stream\_paused = true      get\_tree().paused = true      $Unpause.show()      $Quit.show()      #pauses everything when pause is pressed  func \_on\_Unpause\_pressed():      Global.button = true      $Pause.show()      Global.pause = 0      $Play.stream\_paused = false      get\_tree().paused = false      $Unpause.hide()      $Quit.hide()      #upauses everything when pause is pressed  func \_on\_Quit\_pressed():      Global.button = true      Global.game = 0      Global.pause = 0      get\_tree().paused = false      #quits game loop  func \_process(\_delta):      if Global.cash>0:          $Label.text = " Score: "+str(Global.score)+"\n Coins: " + str(Global.cash)      #Sets score and cash      else:          $Label.text = " Score: "+str(Global.score)      #sets score      if Global.typeShot == 1:          $TypePic.frame = 0          $TypeText.text = "Normal"          #sets img to normal shells      if Global.typeShot == 2:          $TypePic.frame = 1          $TypeText.text = "Incendiary"          #sets img to Incendiary shells      if Global.typeShot == 3:          $TypePic.frame = 2          $TypeText.text = "Large"          #sets img to large shells      if Hudtype == 1:          $HUD.text = " HP: " + str(Global.hp) + " of "+str(Global.userInfo.getMaxHP()) +"\nAmmo: "+ str(Global.ammo) + " of "+ str(Global.userInfo.getMaxShell())          $ProHUDHP.hide()          $ProHUDAmmo.hide()          #sets HUD values      elif Hudtype == 2:          $HUD.hide()          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)          #sets HUD values      else:          $HUD.text = " HP: " + str(Global.hp) + " of "+str(Global.userInfo.getMaxHP()) +"\n Ammo: "+ str(Global.ammo) + " of "+ str(Global.userInfo.getMaxShell())          $ProHUDHP.set\_value(Global.hp)          $ProHUDAmmo.set\_value(Global.ammo)          #sets HUD values      if Global.hp <= 0:          Global.hp = 0          $Play.playing = false          if Global.pause == 0:              $Play.playing = false              if str(Global.userInfo.getVolume()) != "-1.#INF":                  $Death.play()              get\_tree().paused = true              $Pause.hide()              changes()              Global.pause = 1              #makes game stop as hp is 0      else:          if Global.boss == true:              $Play.stream\_paused = true              #pauses music as boss is on screen          else:              if Global.pause != 1:                  $Play.stream\_paused = false                  #unpauses music    func \_on\_Menu\_pressed():      Global.button = true      Global.game = 0      get\_tree().paused = false      self.queue\_free()      #loads munu  func \_on\_Restart\_pressed():      Global.button = true      Global.newGame = 1      get\_tree().paused = false      self.queue\_free()      #restarts game loop  func changes():      if Global.pause == 0:          Global.pause = 1          $Over.show()          var change = false          scores.append(Global.score)          Global.userInfo.setCash(Global.userInfo.getCash()+Global.cash)          Global.cash = 0          $Over/TotalCash.text = "Total cash is " + str(Global.userInfo.getCash())          for i in range(scores.size()-1, -1, -1):              for j in range(1,i+1,1):                  var notes = scores[j]                  var prevnotes = scores[j-1]                  if notes > prevnotes:                      var temp = scores[j-1]                      scores[j-1] = scores[j]                      scores[j] = temp                      change = true          for i in len(scores):              $Over/Scores.text = $Over/Scores.text + str(scores[i]) + "\n"          if change == true:              Global.userInfo.setH1(str(scores[0]))              Global.userInfo.setH2(str(scores[1]))              Global.userInfo.setH3(str(scores[2]))              Global.userInfo.setH4(str(scores[3]))              Global.userInfo.setH5(str(scores[4]))          Global.save\_changes()      #saves changes  func \_input(event):      if event.is\_action\_pressed("ui\_1"):          Global.typeShot = 1      if event.is\_action\_pressed("ui\_2"):          Global.typeShot = 2      if event.is\_action\_pressed("ui\_3"):          Global.typeShot = 3      if event.is\_action\_pressed("ui\_cancel"):          \_on\_Pause\_pressed()      #chanegs ammo type  func \_on\_Play\_finished():      $Play.play()      pass # Replace with function body.      #replays music  func \_on\_Death\_finished():      $End.play()      #plays death music  func \_on\_End\_finished():      $End.play()      #replays death music |
| Nodes |
| Scenes |

### Player.tscn

|  |
| --- |
| extends KinematicBody2D  const Shell = preload("res://Scenes/Shell.tscn")  const Hook = preload("res://Scenes/Hook.tscn")  const ShellFire = preload("res://Scenes/ShellFire.tscn")  const ShellLarge = preload("res://Scenes/ShellLarge.tscn")  #loads scenes that instanced from here  export (float) var rotation\_speed  #makes the multiplier for rotation\_speed  var rotation\_dir = 0  #makes variable for rotation of player  var tleft = ""  var tright = ""  #variable that sets keybind for rotation  var tfire = ""  #variable that sets keybind to fire shells  var thook = ""  #variable that sets keybind to fire hook  var mouseLook = false  #variable that tells if user is using mouse to rotate  var target = Vector2()  #sets its location  var collision  #makes a collision box  onready var main = get\_tree().current\_scene  #gets game scene to instance items to  func \_ready():      rotation\_speed = Global.userInfo.getSens()\*5      $Blue.disabled = true      $Red.disabled = true      $Green.disabled = true      #switches of all collisions boxes off      $Tank\_dark.animation = "default"      #sets sprite to tanks sprites      $Tank\_dark.frame = Global.userInfo.getColour()      #sets colour of tank to users selected      $Scope.frame = Global.userInfo.getColour()      #sets scope to colour of tank      $Tank\_dark.playing = false      #makes it not play animation      $Tank\_dark.show()      #makes tank visible      $Scope.show()      #makes scope visible      if Global.userInfo.getColour() == 0:          $Green.disabled = false      elif Global.userInfo.getColour() == 1:          $Blue.disabled = false      elif Global.userInfo.getColour() == 2:          $Blue.disabled = false      elif Global.userInfo.getColour() == 3:          $Green.disabled = false      elif Global.userInfo.getColour() == 4:          $Red.disabled = false      elif Global.userInfo.getColour() == 5:          $Red.disabled = false      elif Global.userInfo.getColour() == 6:          $Red.disabled = false      elif Global.userInfo.getColour() == 7:          $Green.disabled = false      elif Global.userInfo.getColour() == 8:          $Blue.disabled = false      #different colour have different tank shapes      #there are 3 tank shapes and this switches on      #the correct one      if Global.userInfo.getRotate() == 1:          tleft = "ui\_right"          tright = "ui\_left"          #sets rotation keybind to left and right arrow keys      elif Global.userInfo.getRotate() == 2:          tright = "ui\_a"          tleft = "ui\_d"          #sets rotation keybind to a and d keys      else:          mouseLook = true          #sets movement to mouse position      if Global.userInfo.getFire() == 3:          tfire = "ui\_r"          #sets keybind to fire shell to r      elif Global.userInfo.getFire() == 2:          tfire = "ui\_space"          #sets keybind to fire shell to space      elif Global.userInfo.getFire() == 1:          tfire = "ui\_f"          #sets keybind to fire shell to f      else:          tfire = "ui\_left\_click"          #sets keybind to fire shell to left click      if Global.userInfo.getHook() == 1:          thook = "ui\_h"          #sets keybind to fire hook to h      elif Global.userInfo.getHook() == 2:          thook = "ui\_shift"          #sets keybind to fire hook to shift      else:          thook = "ui\_right\_click"          #sets keybind to fire hook to right click  func \_process(delta):      rotation\_dir = 0      #sets change to 0      if Global.hp <= 0:          if Global.pause == 0:              $Scope.hide()              #hides scope when player dies              $Tank\_dark.animation = "blow"              #sets animation to blowing up              $Tank\_dark.frame = 0              #sets animation to start              $Tank\_dark.playing = true              #makes animation play      if mouseLook == false:          if Input.is\_action\_pressed(tleft):              rotation\_dir += 1          if Input.is\_action\_pressed(tright):              rotation\_dir -= 1          self.rotation += rotation\_dir \* rotation\_speed \* delta          #rotates player on movement      else:          if Global.pause == 0:              look\_at(get\_global\_mouse\_position())              #makes player look at mouse              $Scope.position = self.get\_local\_mouse\_position()              #makes scope position be off the mouse              Input.set\_mouse\_mode(Input.MOUSE\_MODE\_HIDDEN)              #hides mouse invisible          else:              Input.set\_mouse\_mode(Input.MOUSE\_MODE\_VISIBLE)              #makes mouse visible  func \_input(event):      if event.is\_action\_pressed(tfire):          if Global.pause == 0:              if Global.firing == 0:                  if Global.typeShot == 1:                      if Global.ammo > 0:                          Global.firing = 1                          var shell = Shell.instance()                          main.add\_child(shell)                          shell.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -1                          $SFXFire.play()                          yield(get\_tree().create\_timer(0.3), "timeout")                          Global.firing = 0                          #fires normal shell                  elif Global.typeShot == 2:                      if Global.ammo > 1:                          Global.firing = 1                          var shellfire = ShellFire.instance()                          main.add\_child(shellfire)                          shellfire.global\_transform = $Muzzle.global\_transform                          Global.ammo = Global.ammo -2                          $SFXFire.play()                          yield(get\_tree().create\_timer(1.0), "timeout")                          Global.firing = 0                          #fires incendiary shell                  elif Global.typeShot == 3:                      if Global.ammo > 2:                          Global.firing = 1                          var shelllarge = ShellLarge.instance()                          main.add\_child(shelllarge)                          end = $Muzzle.global\_transform                          Global.ammo = Global.ammo -3                          $SFXFire.play()                          yield(get\_tree().create\_timer(0.6), "timeout")                          Global.firing = 0                          #fire large shell      if event.is\_action\_pressed(thook):          if Global.pause == 0:              if Global.hook == 0:                  Global.hook = 1                  var hook = Hook.instance()                  main.add\_child(hook)                  hook.global\_transform = $Muzzle.global\_transform                  #fires hook  func \_on\_Tank\_dark\_animation\_finished():      $Tank\_dark.hide()      #hides tank after explosion |
| Nodes |
| Scene |

### Zombie.tscn

|  |
| --- |
| extends KinematicBody2D  var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  var hit = false  var target = Vector2(990, 600)  #gets variable ready  func \_ready():      rng.randomize()      #makes zombie animation play      var where = rng.randi\_range(0,1) # 0 means on base 1 = on sides      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              var along =  rng.randi\_range(0,1080)              position.y = along          else:              position.x = 1980              var along =  rng.randi\_range(0,1080)              position.y = along      else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              var along =  rng.randi\_range(0,1980)              position.x = along          else:              position.y = 1080              var along =  rng.randi\_range(0,1980)              position.x = along      #sets position of zombie      $Sprite.modulate = Color(rng.randf\_range(0.5,1),rng.randf\_range(0.5,1),rng.randf\_range(0.5,1))      #modulates zombies colour  func \_physics\_process(\_delta):      look\_at(target)      #makes it look at player      var inc = Global.ticks/5      #increases speed at number of game ticks      velocity = position.direction\_to(target)\*inc      #plans on where it moves this cycle      if Global.game == 0:          self.queue\_free()          #self-terminate if game ends      if Global.hp <= 0:          self.queue\_free()          #self-terminate if game if player dies      if Global.pause == 1:          $Sprite.playing= false          #pauses animation as game is paused          if Global.newGame == 1:              self.queue\_free()              #self-terminate if game resets          pass      elif position.distance\_to(target) > 5:          collision = move\_and\_collide(velocity)          #moves player and finds collisions      if Global.pause == 0:          $Sprite.playing= true          #keeps animation playing          if collision:              var item = str(collision.collider.name)              item = item.replace("@","")              item = item.replace("1","")              item = item.replace("2","")              item = item.replace("3","")              item = item.replace("4","")              item = item.replace("5","")              item = item.replace("6","")              item = item.replace("7","")              item = item.replace("8","")              item = item.replace("9","")              item = item.replace("0","")              #removes  any metadata from colliders name              if hit ==false:                  if item == "Shell":                      hit = true                      Global.score = Global.score + int(Global.ticks)                      Global.cash = Global.cash +1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXDie.play()                      yield(get\_tree().create\_timer(0.45), "timeout")                      self.queue\_free()                      #increases score and cash then                      #then plays death sfx then self terminates                  elif item == "Tank":                      hit = true                      Global.hp = Global.hp -1                      Global.zombieNum = Global.zombieNum-1                      self.hide()                      self.position.x =0                      self.position.y =0                      $SFXEat.play()                      yield(get\_tree().create\_timer(1.3), "timeout")                      self.queue\_free()                      #decrease hp by 1 then plays eat sfx then self terminates |
| Nodes |
| Scene |

### BossZombie.tscn

|  |
| --- |
| extends KinematicBody2D  var velocity = Vector2()  var collision  var rng = RandomNumberGenerator.new()  var hp = Global.score  var yes = true  var target = Vector2(990, 600)  #gets variable ready  func \_ready():      $ZombieHP.set\_max(hp)      #sets health  on current score.        if str(Global.userInfo.getVolume()) != "-1.#INF":          $Battle.play()      #plays boss music      rng.randomize()      $Sprite.playing= true      var where = rng.randi\_range(0,1) # 0 means on base 1 = on sides      if where == 1:          var side =  rng.randi\_range(0,1) # 0 means on left 1 = on right          if side == 0:              position.y = rng.randi\_range(100,980)              position.x = 92          else:              position.x = 1880              position.y = rng.randi\_range(100,980)      else:          var side =  rng.randi\_range(0,1) # 0 means on top 1 = on bottom          if side == 0:              position.x = rng.randi\_range(100,1880)              position.y = 100          else:              position.y = 980              position.x = rng.randi\_range(100,1990)      #sets position of zombie      $Sprite.modulate = Color(rng.randf\_range(0.5,1),rng.randf\_range(0.5,1),rng.randf\_range(0.5,1))      #modulate colour of mob to have slight variations.  func \_physics\_process(\_delta):      if hp <= 0:          Global.score = Global.score +30 + int(Global.ticks)          Global.cash = Global.cash +30 + int(Global.ticks)          Global.boss = false          self.queue\_free()      #if it dies it increases score and cash.      look\_at(target)      velocity = position.direction\_to(target)\*0.25      #goes to the target.      if Global.hp <= 0:          self.queue\_free()      #if game ends or player dies it self-terminates.      if Global.pause == 1:          $Sprite.playing= false          $Battle.stream\_paused = true          pass      #pasues zombie if game pauses      if Global.pause == 0:          $Battle.stream\_paused = false          $Sprite.playing= true          #keeps music playing and sprite animated          collision = move\_and\_collide(velocity)          #moves zombie          $ZombieHP.set\_value(hp)          if collision:              var item = str(collision.collider.name)              if item == "Shell":                  if yes == true:                      yes = false                      hp = hp-1                      yield(get\_tree().create\_timer(0.01), "timeout")                      yes = true                      #if it hits the shell the hp decreases by 1 every 0.01s              elif item == "Tank":                  Global.hp = Global.hp -100                  self.hide()                  self.position.x =0                  self.position.y =0                  Global.boss = false                  self.queue\_free()                  #if it hits the tank decreases hp by 100  func \_on\_Battle\_finished():          $Battle.play()          #if battle music has stopped makes it play again |
| Nodes |
| Scene |

### MedKit.tscn

|  |
| --- |
| extends KinematicBody2D  var hit =0  var rng = RandomNumberGenerator.new()  #gets variables ready  func \_ready():      rng.randomize()      self.position.x = rng.randi\_range(100,1880)      self.position.y = rng.randi\_range(100,980)      #randomly places medkit on screen      yield(get\_tree().create\_timer(10.0), "timeout")      Global.mKit = 0      self.queue\_free()      #self terminates scene after 10 seconds.  func \_process(\_delta):      if Global.hp <= 0:          self.queue\_free()      #sees if medkit shouldn't exist then self terminates      if hit == 0:          if Global.mKit == 0:              hit = 1              self.hide()              $Get.play()              yield(get\_tree().create\_timer(0.76), "timeout")              self.queue\_free()      #sees if it has been hit then plays sfx the self terminates |
| Nodes |
| Scene |

### AmmoKit.tscn

|  |
| --- |
| extends KinematicBody2D  var hit =false  var rng = RandomNumberGenerator.new()  #gets variables ready  func \_ready():      rng.randomize()      self.position.x = rng.randi\_range(100,1880)      self.position.y = rng.randi\_range(100,980)      #randomly places ammo kit onto the screen      yield(get\_tree().create\_timer(10.0), "timeout")      Global.aKit = 0      self.queue\_free()      #self terminates scene after 10 seconds.  func \_process(\_delta):      if Global.hp <= 0:          self.queue\_free()      #sees if ammokits are supposed to be there      #self-terminates if false.      if hit == false:          if Global.aKit == 0:              hit = true              self.hide()              $Get.play()              yield(get\_tree().create\_timer(0.5), "timeout")              self.queue\_free()      #sees if it has been hit then plays sfx the self terminates |
| Nodes |
| Scene |

### Hook.tscn

|  |
| --- |
| extends KinematicBody2D  export (int) var speed = -100  var velocity = Vector2()  var collision  #stes speed and variables  func \_process(delta):      velocity += transform.y \* speed      collision = move\_and\_collide(velocity \* delta)      #moves hook      if collision:          if str(collision.collider.name) == "MedKit":              Global.hp = Global.hp + Global.userInfo.getkitHP()              if Global.hp > Global.userInfo.getMaxHP():                  Global.hp = Global.userInfo.getMaxHP()              Global.hook = 0              Global.mKit = 0              self.queue\_free()              #if it hits medkit is increases hp and self terminates          if str(collision.collider.name) == "AmmoKit":              Global.ammo = Global.ammo + Global.userInfo.getkitAmmo()              if Global.ammo > Global.userInfo.getMaxShell():                  Global.ammo = Global.userInfo.getMaxShell()              Global.hook = 0              Global.aKit = 0              self.queue\_free()              #if it hits ammokit is increases hp and self-terminates  func \_on\_Out\_screen\_exited():      Global.hook = 0      self.queue\_free()      #if it goes out of the screen it self-terminates |
| Nodes |
| Scene |

### Shell.tscn

|  |
| --- |
| extends KinematicBody2D  export (int) var speed = -100  var hit = false  var playing = false  var velocity = Vector2()  var collision  #sets variables  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.show()      $Sprite.hide()      #resets everything  func \_physics\_process(delta):      velocity += transform.y \* speed      if Global.pause == 1:          pass          #makes it stop movement if game is paused      elif hit == false:          collision = move\_and\_collide(velocity \* delta)          if collision:              if str(collision.collider.name) == "Zombie":                  hit = true      else:          if playing == false:              playing = true              $Bullet.hide()              $Sprite.frame = 0              $Sprite.show()              $Sprite.playing = true              #hides bullet and makes explosion animation play              $CollisionShape2D.scale.x = 4              $CollisionShape2D.scale.y = 4              #makes explosion bigger  func \_on\_Sprite\_animation\_finished():      self.queue\_free()      #once animation stops it self-terminates  func \_on\_Out\_screen\_exited():      self.queue\_free()      #once it outside window it self-terminates |
| Nodes |
| Scene |

### ShellFire.tscn

|  |
| --- |
| extends KinematicBody2D  export (int) var speed = -100  var hit = false  var playing = false  var velocity = Vector2()  var collision  #sets variables  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.show()      $Sprite.hide()      #resets everything  func \_physics\_process(delta):      velocity += transform.y \* speed      if Global.pause == 1:          pass          #makes it stop movement if game is paused      elif hit == false:          collision = move\_and\_collide(velocity \* delta)          if collision:              if str(collision.collider.name) == "Zombie":                  hit = true      else:          if playing == false:              playing = true              $Bullet.hide()              $Sprite.frame = 0              $Sprite.show()              $Sprite.playing = true              #hides bullet and makes explosion animation play  func \_on\_Sprite\_animation\_finished():      self.queue\_free()      #once animation stops it self-terminates  func \_on\_Out\_screen\_exited():      self.queue\_free()      #once it outside window it self-terminates |
| Nodes  A picture containing text, meter, device  Description automatically generated |
| Scene |

### ShellLarge.tscn

|  |
| --- |
| extends KinematicBody2D  export (int) var speed = -100  var hit = false  var playing = false  var velocity = Vector2()  var collision  #sets variables  func ready():      $CollisionShape2D.scale.x = 2      $CollisionShape2D.scale.y = 2      $Bullet.show()      $Sprite.hide()      #resets everything  func \_physics\_process(delta):      velocity += transform.y \* speed      if Global.pause == 1:          pass          #makes it stop movement if game is paused      elif hit == false:          collision = move\_and\_collide(velocity \* delta)          if collision:              if str(collision.collider.name) == "Zombie":                  hit = true      else:          if playing == false:              playing = true              $Bullet.hide()              $Sprite.frame = 0              $Sprite.show()              $Sprite.playing = true              #hides bullet and makes explosion animation play              $CollisionShape2D.scale.x = 8              $CollisionShape2D.scale.y = 8              #makes explosion bigger    func \_on\_Sprite\_animation\_finished():      self.queue\_free()      #once animation stops it self-terminates  func \_on\_Out\_screen\_exited():      self.queue\_free()      #once it outside window it self-terminates |
| Nodes  A picture containing text, meter, device  Description automatically generated |
| Scene    A screenshot of a video game  Description automatically generated |

## Video Evidence

### Login

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=77> | The program first gets a signal from button presses to do 1 of 3 things: Quit the program, open the registration scene or login.  The quit just closes the program and the registration makes Main instance the registration scene and for Login to close itself.  The login button will first make all the text boxes uneditable then get the username and password from them. The program then sees if the username exists in the database then it hashes the password and gets the index from the username then checks to see if the password is in the hash table. If it is then it changes the singleton variable logged to true showing that a user is logged in, then it hands the users data to the user class to instance in the globals.gd. It then tells the main scene via the globals.gd variable to open the menu scene and then Login to close itself. If any of the data does not match to what is in the database, it will show a label telling there was none found then empty the text boxes and then makes them editable again. |

### Registration

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA> | The program gets 3 signals: quit, back to login or register. Quit closes the whole program. Back to login will tell the main scene to instance the login scene then registration will self-terminate.  If register launches it will first make check if the username already exists, then finds checks if the username is the minimum character length (3) it will then check if the password meets the requirements. It will then hash the password and then indexes the username then it will store the username in userData.dat file then it will store the hashed password in hashPass.dat. It will then tell main to open login then the scene will terminate.  If even one of the requirements is not met it will open an error dialog telling the user what is at fault. |

### Menu

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA> | The menu scene will show the current users cash and then gives button to play. There is a quit button and logout button, they will launch a dialog and that asks for permission to do it, the quit will close the program and logout will turn the globals.gd variable logged into false then tell the main scene to instance login then self-terminate.  The start button will tell the main scene to instance the Game scene then menu will self-terminate.  The shop button will tell the main scene to instance the Shop scene then menu will self-terminate.  The scores button will tell the main scene to instance the Scores scene then menu will self-terminate.  The controls button will tell the main scene to instance the Controls scene then menu will self-terminate.  The settings button will tell the main scene to instance the Settings scene then menu will self-terminate.  The lore button will tell the main scene to instance the Lore scene then menu will self-terminate.  The credits button will tell the main scene to instance the Credits scene then menu will self-terminate. |

### Shop

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=471> | Once open it will tell user their current amount of cash they have. Then it will load the text on what the buttons will do and how they will affect the user. The program will then wait until 1 of 5 buttons is pressed: back to menu, upgrade max HP, upgrade max HP regen, upgrade max ammo, or upgrade max ammo regen.  If back to menu is pressed, then the main scene would the load the menu scene then the shop would self-terminate.  If the upgrade is pressed first it checks if the price is lesser than the current money of user then changes attribute of object then save to the file and if it fails then it will tell the user that they have insufficient money. |

### Scores

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=194>  <https://youtu.be/3R96poKCVDA?t=665> | The User is shown all their scores and the current high scores of all the other user in the current file. They are also given a button that causes the main scene to instance the menu and then self-terminate. |

### Controls

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=167> | First it gets the creates the text box by finding the users options and then using a set text to concatenate a string to put into the text box.  Then it starts by telling the controls by using premade soundtracks and plays the correct one for the user.  It also has button that causes the main scene to instance the menu and then self-terminate the scene. |

### Credits

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=163> | It has button that causes the main scene to instance the menu and then self-terminate the scene. |

### Lore

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=225> | As soon it is loaded it uses the video player node to play the lore video. It has a button to go back to menu or to skip to the end. The back button causes the main scene to instance the menu and self-terminates the scene. The skip causes the video to end to show the holdout picture. The end hides the skip and shows the replay button.  The back button causes the main scene to instance the menu and then self-terminate the scene. |

### Settings

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=105>  <https://youtu.be/3R96poKCVDA?t=642> | It allows the toggling of fullscreen. It allows the changing of volume which changes the attribute in the object but doesn’t save until the save and back to menu button is pressed. It allows the resetting of user data to what it was at the start. The save and back button causes the program to save all user data to the file and then causes the main scene to instance the menu and then self-terminate the scene. It allows the editing of the tank colour of the user. It allows the allows the changing of the controls on how the user rotates, fires the shell and hook. It does this by changing the attribute of the object. |

### Delete

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=657> | It waits until 1 of 2 buttons is pressed: back to menu or delete.  The back button causes the main scene to instance the menu and then self-terminate the scene.  The delete button will make the text boxes uneditable then check if the username entered is the same username of the user, it will then check is the username and password is correct. Once it done all of this it then expunges the user’s data from the file and turns their hashed password in the hash table to ‘del’, it will then automatically logout. If any of this fails it will tell the user, the fault. |

### Game

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=353> | Once it is instanced it will instance the player, the HUD then start instancing the zombies while controlling the game ticks. Every 1 minute it instances the boss and it will instance a med or ammo kit every 10 seconds if they are below max. It will self-terminate if the singleton variable game is 0, causing all children scenes (e.g. player, HUD, zombies, bosses, ammo kits and med kits) to also self-terminate. |

### HUD

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=353> | Allows the user to see current health, ammo, ammo type, score, and cash. It handles the games music. It also has a button to pause the game and quit to menu. It also handles game over, when health is equal to 0 it will cause the game to pause, player to hide, shows the scores and the game over music plays. |

### Player

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=353> | If the user rotates using their keybind then the player rotates to desired rotation. It will instance the shells when the appropriate keybind is pressed, same with the hook. Makes the explosion animation play if the hp is 0. |

### Zombie

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=353> | It will look at go to the centre of the screen will it touches a shell, or it touches the tank. If it touches the tank it reduces the singleton variable hp by 1, plays the eating sfx the self-terminates. If it touches a shell it increases cash by 1 and then increases the score by the current game ticks rounded to the nearest whole number. If the hp equals 0 it will self-terminate and if the game pauses it will not move. |

### BossZombie

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA?t=412> | It will look at go to the centre of the screen will it touches a shell, or it touches the tank. If it touches the tank it reduces the singleton variable hp by 100, plays the eating sfx the self-terminates. If its health equals 0 it will increase cash by 30 and the ticks to the nearest whole number. Then increases the score by the current game ticks rounded to the nearest whole number and 30. If the singleton variable hp equals 0 it will self-terminate and if the game pauses it will not move. |

### Main

|  |  |
| --- | --- |
| https://youtu.be/3R96poKCVDA | It instances most scenes, and controls menu music and makes sure that there is a hash table to be used. Also handles the background and main volume control. |

### globals

|  |  |
| --- | --- |
| <https://youtu.be/3R96poKCVDA> | It is a singleton that I use to store my class, object, variables and functions that that the program uses often. |

# Bibliography

|  |  |
| --- | --- |
| What | Where |
| Making the timer using yield(get\_tree().create\_timer(2.0), "timeout")” | <https://gdscript.com/solutions/godot-timing-tutorial/> |
| Collisions | <https://docs.godotengine.org/en/stable/tutorials/physics/physics_introduction.html> |
| Instancing | Mr. Ngai |
| How to get denary of a character | <https://docs.godotengine.org/en/stable/classes/class_@gdscript.html>? |
| How to use the autoloader/make singleton variables | <https://www.youtube.com/watch?v=5k8xuefBQHs&ab_channel=Davies> |
| Close the game | <https://godotengine.org/qa/554/is-there-a-way-to-close-a-game-using-gdscript> |
| Random numbers | <https://docs.godotengine.org/en/stable/classes/class_randomnumbergenerator.html> |
| How to read and write files | <https://docs.godotengine.org/en/stable/classes/class_file.html?highlight=file> |
| Rotation | <https://docs.godotengine.org/en/stable/tutorials/2d/2d_movement.html> |
| Audio control | <https://docs.godotengine.org/en/stable/tutorials/audio/audio_streams.html> |
| Audio control | <https://www.gdquest.com/tutorial/godot/audio/volume-slider/> |
| Classes | <https://godotengine.org/qa/9830/handling-classes-in-godot> |
| Video | <https://docs.godotengine.org/en/stable/classes/class_videoplayer.html> |