A Level Project

Name: Maxim Ladoshin

Centre:

Candidate Number:

Contents

[Analysis 3](#_Toc58567207)

[Introduction 3](#_Toc58567208)

[Computational Suitability 3](#_Toc58567209)

[Research 3](#_Toc58567210)

[Stakeholders 3](#_Toc58567211)

[Interviews 3](#_Toc58567212)

[Interview 1: 3](#_Toc58567213)

[Interview 2 3](#_Toc58567214)

[Interview Conclusion 3](#_Toc58567215)

[Requirements 3](#_Toc58567216)

[Hardware Requirements 4](#_Toc58567217)

[Software Requirements 4](#_Toc58567218)

[Design 5](#_Toc58567219)

[Interface 5](#_Toc58567220)

[Variables 5](#_Toc58567221)

[Functionality 5](#_Toc58567222)

[Test Plan 5](#_Toc58567223)

[Implementation 6](#_Toc58567224)

[Iteration 1 6](#_Toc58567225)

[Requirements being developed 6](#_Toc58567226)

[Errors 6](#_Toc58567227)

[Conclusion 6](#_Toc58567228)

[Iteration 2 6](#_Toc58567229)

[Requirements being developed 6](#_Toc58567230)

[Errors 6](#_Toc58567231)

[Conclusion 6](#_Toc58567232)

[Iteration 3 6](#_Toc58567233)

[Requirements being developed 6](#_Toc58567234)

[Errors 6](#_Toc58567235)

[Conclusion 6](#_Toc58567236)

[Testing 7](#_Toc58567237)

[Evaluation 8](#_Toc58567238)

# Analysis

## Introduction

Nowadays the game industry becomes more and more popular amongst people of all ages. Computer games have been historically the type of entertainment when people can forget about all their problems in real life and dive into the game reality. For some people games have become a process of relaxation after a hard day, or just a sort of funny entertainment when they are bored (e.g. while travelling in a bus, train etc). Moreover, some games can develop person’s mind, improve his reaction and cognitive skills which is sometimes quiet useful for small children. Another games are used to educate children and teach the everything about the world (Animal Jam – zoology, Endless Alphabet – alphabet, Starfall Kids Games). There is an enormous amount of various types of games accessible today, which can educate you, develop some of your skills or just entertain you.

Rapid technology changes in mobile devices have revealed the whole new mobile gaming platform, which now is gaining more and more new players due to its convenience and cheap price. Almost every person in the world have access to a mobile phone, so mobile games are accessible to an extremely wide audience.

Most of the popular games in 21 century try to reach the maximum possible realism, using quiet a lot of hardware resources. Every year there is a new graphic technology invented, which improves the games’ graphic such as 3D, Ray Tracing, DLSS and so on and so forth. All this technologies make the games really close to the real life. However, less and less people can afford to play this games each year because such AAA games require an expensive hardware (personal computers or laptops). Such high-end games as Cyberpunk, GTA 5, Red Dead Redemption 2 are one of the most graphic demanding games out now (in 2021). Moreover, the development of such high-end games is quiet costly, which make their final price in the market higher. Therefore, the gaming industry tends to be more and more expensive for the players.

Besides that, the other problem of modern games is that they require the internet connection which is not always the case. For example, if you have a long 10 hour flight and you are very bored, you can’t play GTA 5, Fortnite or any other multiplayer game. So, the only choice is to play arcade games or any other game which doesn’t require an internet connection.

I personally like RPG and strategy games such as Age of Empire, GTA 2 and Terraria. These games are relatively simple and straight forward in terms of graphics as they are all 2D games. Even though, I enjoyed these games so much in childhood. These games are extremely profound and exciting to play.

That is why I decided to develop a game which will combine some elements from arcade 2D games, shooter games and RPG games. My game will also not require any internet connection, so it can be played anywhere. In addition, my game will be suitable for low-end devices, so more stakeholders could appreciate it. The primary input devices will be the keyboard and a mouse/touchpad.

## 

## Computational Suitability

The problem I will be solving will require to use computer calculations such as checking for collisions, player’s health, enemies’ health and so on. The game I will develop is an RPG game with the elements of a shooter, so it obviously cannot be played in real life. This game would provide a virtual world where you will be battling against enemies, searching for new Weapons, armor, medicine kits and so forth.

Moreover, the computational power of a device allows me to implement enemy AI, so the enemies won’t just rest at one place, but try to attack the player. The enemy AI will make a game more realistic and more fun. It will be more difficult to survive in a game where all the enemies are trying to terminate you meaning the user will have to come up with various strategies to cope with enemies (such as hiding, running away, killing them etc). Finally, the implementation of a simple AI will make the enemies similar to real life players, which will definitely make a game more exciting to play.

Trying to solve such a problem in a real-life world is impossible as you obviously have a virtual map with obstacles and randomly located enemies. The number of enemies will increase over time as well as the player’s score and number of kills.

* Encapsulation may help me to develop a code which will be more secure, and players will have to try considerably harder to cheat during the game. Furthermore, I could use encapsulation to encourage other programmers or users to modify the source code and add new feature. In this case, encapsulation will make sure that class properties are not accessed or changed in the way they should not be. This will prevent program crashes and make the development process clearer and pleasing.
* Abstraction will help me to develop my game without huge troubles by focusing on core functionality of my game and excluding all the unessential details which will only make the code longer and more complicated. In my game I will use pygame library which is already pre-build to help python programmers build games and significantly reduce the amount of code. The programmer does not really have to know how the pygame methods work to create a game. he just only needs to know how to use these methods.
* Data Visualization

My game will use different data such as score, player’s health, screen size constant, color constants, player’s inventory list, and so on. The GUI (graphic user interface) will be rendered according to all this data and it will allow the stakeholders to better navigate and control inside the game, which will make my game user-friendly.

## 

## Research

I was inspired to develop my own 2D RPG game by such games as “Enter The Gungeon”, “Nuclear Throne”, and “Grand Theft Auto 2”. All these games have a decent UI which is very comfortable and easy to use.

* **Enter the Gungeon**

Enter the Gungeon is a single/cooperation 2D RPG shooter game. Player should kill the enemies and complete the levels to reach the final prize – the weapon that can demolish the past. This game includes different types of weapons, characters, enemies, NPCs and so on. The user should look for better ammunition that can be found in different places as a loot. The final mission is to kill the main boss and the player will be awarded a main prize as he has done that successfully.

A picture containing text, indoor

Description automatically generated

Figure 1 - Enter the Gungeon - game screenshot

**A screen shot of a video game

Description automatically generated with medium confidence**

Figure 2 - "Enter The Gungeon" - game screenshot

**I was inspired by this game because:**

* Enter the Gungeon has a great UI design.
* The game has lots of content with different bosses, enemies, NPCs and other cool stuff such as various weapons, spells, types of armor etc.
* The game has nice animations and is smooth.
* It reminds me of retro 2d arcades, though it is a modern and refined version of old games.
* **Nuclear Throne**

Nuclear Throne is a top-down shooter game which is very dynamic and intensive. There is so many things going on in the game, so you become a bit lost in the beginning but get used to it a bit later. The game mimics retro 2D arcades, although it has improved design and looks quite nice.

Map

Description automatically generated

Figure 3 - "Nuclear Throne" - game screenshot

The player in “Nuclear Throne” should always be moving and cope with all types of enemies, such as mice, monsters, mutants and bosses. The game also includes elements of an RPG such as looting, character customization and dialogs with NPCs which can help you progress faster in the virtual world.

Map

Description automatically generated

I was inspired by this game because it has character customization, in-game progression, different difficulty levels and advanced enemy AI which makes this game more realistic and fun to play. I would like to add some of game’s features into my own game. For instance, combat system, looting systems, some UI elements (health bar, menus)

Figure 4 - "Nuclear Throne" - game screenshot

## Grand Theft Auto 2 (GTA 2)

Almost everyone knows about GTA games. These are the open-world games, where the player have minimum limitations. This game is a virtual world where you can do whatever you want. This fact seems quite appealing to me. GTA series started with top down 2D games and the became one of the best 3D AAA games (GTA 5). Even though this game was launched in 1991, it still astonishes me and I enjoy spending some time playing this legendary game.

The things I like about GTA 2 are:

* The game provide an open world with NPCs and enemies
* The player has lots of choice
* The game includes elements from RPG, shooter and vehicle driving games.

## A picture containing text, control panel Description automatically generated

Figure 5 - "Grand Theft Auto 2" - game screenshot

Grand Theft Auto 2 had one of the best graphics at that time in 1999 and I quiet like it. That’s why it would be ideal if I could implement similar design in my game. Moreover, I enjoy this game because it has lots of freedom for a player – players can make and spend money, drive different vehicles, shoot enemies, run away from the police, complete missions and so many more. This wide variety of options makes the game more interesting and satisfying to play.

## Stakeholders

My game is going to be a simple indie arcade which will provide the player with a wide variety of paths to progress in game (such as get better weapons, armor). So, my stakeholders will be people who like playing arcade games, top-down shooters, or retro 2D RPG games. Moreover, most of the people who doesn’t have a stable internet connection would appreciate my game, because it won’t require any internet to play it. Besides that, the owners of low-end devices can benefit from my game because it will suit their low performance devices and they will be able to play it without lags.

Finally, the students in the boarding house will appreciate this game because sometimes the internet in the boarding house goes down and these guys don’t know how to entertain themselves.

However the target audience for 2D retro game is not wide, I could add some modern features in it such as refined and new-looking graphics and textures, better ambient sound etc. Besides that, it will make my game to stand out from other arcades if I distribute my game for free. This implies, more people will afford it and play. Even If they don’t like it, they won’t regret because they haven’t paid a penny for it. I guess this game won’t suit most of the players who only values astonishing 3D graphics in games and plays only AAA games such as GTA5, Cyberpunk and Call Of Duty.

Who can help you design the game?

To help me with the game’s design I can ask my friends from a boarding house. Some of them have designed games before, so they can give me a helpful device. Moreover, I could ask computer science teachers to help me with game level design and maybe UI design. Finally, my family may help me with designing my game. They could choose the best design for in-game elements (such as textures and objects)

# Interviews

I have surveyed several friends of mine in the boarding house and my parents.

I have collected some useful responses that helped me design the game and its structure to suit the most of my stakeholders.

I decided to survey 3 people with different interests and different ages to get unbiased results.

I have interviewed:

1. Shiwei Hangs (16 years old) – he is a boarder student in my school and he is passionate about physics. He loves computer games, however he is interested in mobile games (such as Subway Surfers and Crossy Road) and AAA games (such as Assasin’s Creed, Cyberpunk)
2. Martin Kozon (19 years old) – hard-working student who is interested in arts and design. Doesn’t mind playing some strategies on his laptop sometimes: Age of Empires
3. Makar Ladoshin (10 years old) – he likes playing games on his tablet. He also plays computer games such as GTA5 and Minecraft on the weekends.

## Interview Questions

Q1: Do you play computer games?

Q2: How often do you play them?

Q3: What is your main gaming device?

Q4: Which types of games do you like the most?

Q5: What are your favorite games?

Q6: Would you try playing 2D RPG shooter game?

Q7: When and where would you most likely play it?

Q8: What sorts of weapons and battle techniques would you like to see in the game?

Q9: Do you want to pick up random loot and to look for rare items in a game?

## Interview 1 (Shiwei Hangs):

Q1: Do you play computer games?

“Yes, I do”

Q2: How often do you play them?

“I usually play in the evenings when I have some free time after school, or on the weekends when I have a lot of spare time. I spend about 2 hours in average playing games a day.”

Q3: What is your main gaming device?

“Laptop and phone”

Q4: Which types of games do you like the most?

“AAA games, Shooters and mobile arcades”

Q5: What are your favorite games?

“Far cry 4, Grand Theft Auto 5”

Q6: Would you try playing 2D RPG shooter game?

“I won’t mind trying”

Q7: When and where would you most likely play it?

“I would most likely play it when I don’t have internet connection, while travelling for example.

Q8: What sorts of weapons and battle techniques would you like to see in the game?

“Perhaps, it will be cool if you can add firearms and some knifes into your game”

Q9: Do you want to pick up random loot and to look for rare items in a game?

“Yes, definitely”

## Interview 2 (Martin Kozon):

Q1: Do you play computer games?

“Yes, I sometimes do”

Q2: How often do you play them?

“I don’t have time to play games during the week, so I usually game during weekends about 2 hour every weekend”

Q3: What is your main gaming device?

“Laptop”

Q4: Which types of games do you like the most?

“Strategy games”

Q5: What are your favorite games?

“Age of Empires, HearthStone”

Q6: Would you try playing 2D RPG shooter game?

“I will try if there are some strategy elements or it is just fun to play.”

Q7: When and where would you most likely play it?

“If I like it I can play it on weekends to relax and have a great time.”

Q8: What sorts of weapons and battle techniques would you like to see in the game?

“I don’t really mind. It can be whatever.”

Q9: Do you want to pick up random loot and to look for rare items in a game?

“Yes, why not. This may make a game more interesting to play.”

## Interview 1 (Makar Ladoshin):

Q1: Do you play computer games?

“Yes”

Q2: How often do you play them?

“I play games every day after school or in the evening to rest after a school day.”

Q3: What is your main gaming device?

“Ipad”

Q4: Which types of games do you like the most?

“RPG shooters, strategy games”

Q5: What are your favorite games?

“Brawl Stars, Clash Royal, Minecraft”

Q6: Would you try playing 2D RPG shooter game?

“I really enjoy playing Brawl Stars, so I am excited to try new 2D RPG games.”

Q7: When and where would you most likely play it?

“I can play it even instead of Brawl Stars and play about 1 hour every evening”

Q8: What sorts of weapons and battle techniques would you like to see in the game?

“I would enjoy various guns and missiles, maybe some sniper rifles. Also, it will be great if there are some different combat styles: hiding, rushing or sneaking”

Q9: Do you want to pick up random loot and to look for rare items in a game?

“Yes, I guess it will be amazing!”

# Interview Conclusion

The interview has given me essential information about how my future game will look like. I will try to reasonably satisfy most demands of my stakeholders in my game.

* most of my stakeholders play computer games quiet often and have access to laptops or mobile devices
* All my stakeholders are willing to try out my game, which is quiet good
* Some of my stakeholders stated they enjoy shooters, others mentioned they like strategies. This means my game should be a combination of shooter and strategy game. So, I could add some elements and techniques from shooter games – shooting, weapons, bullets, inventory etc. And I could add some elements of a strategy game – various paths to kill enemies and get better loot or even add some missions.
* Most of my stakeholders said they would use the game during rest time after school, so I should make the game simple and the UI – user-friendly. If the UI is simple to use, stakeholders won’t be distracted from the gaming process.
* Most of my stakeholders have already played games on a pc, so I can easily use keyboard and a mouse as a primary control devices for this game.

All in all, according to the results I have received, it is reasonable to make a 2D top-down shooter with elements of RPG game. Moreover, the player should be able to pick up the loot and use different types of weapons to battle the enemies. This type of game would be relatively popular among people that I have interviewed.

# Requirements (Success Criteria)

List the requirements of you game.

Must have:

1. Screen size = 800x600 pixels

*The screen size may be changed in the code (change the constant)*

1. Destructible and non-destructible walls

*The player and enemies can destruct the walls and shoot through the holes in walls.*

1. Player sprite

*Use sprite for easier collision detection with bullets, wall bricks and enemies.*

1. Enemies with some sort of AI (chasing the player, shooting etc.)
2. Player must move in 4 directions.

*Up, Down, Left, and right. Moreover, the user can press the different keys simultaneously and move diagonally. This is the most suitable type of movement for this game.*

1. Player can shoot.

*Player can click the right mouse button and release the bullets from the weapon.*

1. Player can pick up the loot.

*Player will be able to collect the loot and store the items of the loot in the inventory.*

1. Player can die.

*If the health of a player is 0 or below, he dies. This will stop the game, so the user can restart the game.*

1. Loser and Winner window
2. Auto generation of loot on the map

*The loot will be randomly generated, so it will be a bit more interesting.*

1. Enemy might have random loot, which the player can get after killing him.
2. Inventory can be full.

*If the weight of the items in the inventory at the limit, the inventory is going to be full, so the player will not be able to pick up the loot. This will make the game more realistic and a bit harder to play.*

1. Adding the items to inventory
2. Displaying the weight of the items in the inventory
3. Player can heal using the medicine kits.
4. Player can use the armor.
5. Armor adds the armor points, which can be drawn after getting the damage from the enemies.
6. 3 types of medicine kits: big (restores 50% of health), medium (restores 20% of health), small (restores 10% of health)
7. 3 types of armor: heavy, medium, light
8. More items the player has in his inventory, the slower he moves.
9. Heavy and medium armor can decrease the player’s speed.
10. Player can pick up the weapons.
11. Player can pick up the bullets.
12. Weapons cannot shoot if there are no bullets.
13. 3 types of enemies:
14. 3 Bosses:
15. First boss with a bow (easy)
16. Second Boss with a sniper rifle (medium)
17. Third boss with a rifle and missiles (hard)
18. Player can select the weapon using buttons 1, 2, 3
19. Player can use other items from inventory using buttons R, T, Y

**Should have:**

1. Close distance combat enemies (armed with knives, katanas, axes)
2. Medium distance combat enemies (armed with pistols, bows or rifles)
3. Far distance combat enemies (armed with sniper rifles)
4. Enemies could shoot the player.

**Could have:**

1. Player can drop the weapon.
2. Player can drop the items in inventory.
3. Player can use spells to increase his speed.
4. The camera follows the player.

**Won’t have:**

1. Internet connection
2. 3D

# 

# Hardware Requirements

1. Screen with minimum size of 800x600 px
2. Keyboard
3. Mouse or touchpad
4. Free space on disk
5. Dual core processor

# Software Requirements

1. Operating system (Windows, Linux, Mac os)
2. Python (version 3.7.6 or later)
3. PyGame library (version 1.9.3 or later)

# Design

## Interface

Hand drawn diagrams of what the game looks like. Label the important aspects.

Basic requirements:

1. 800x600px window
2. Player moves in all 4 directions (up, down, left, and right)
3. The player moves when the keys WASD are pressed
4. The player can boost their speed if the shift key is pressed
5. The player can shoot if the left mouse button is clicked
6. The player can select weapons using keys 1, 2 or 3.

Interface design:

**800px**

**600px**

*Player can move in four directions up, down right, and left.*

## Variables and Data Structures

List all classes and methods clearly.

Inheritance diagram.

The game generates the map, consisting of blocks 16x16 pixels. The player is a class with its own functionality – methods. There is also an enemy class with its unique methods. Enemy class and player class are both children of a parent class Person which has basic methods like move, shoot etc.

**Global Variables:**

Colors:

BLACK = (0,0, 0)

WHITE = (255, 255,255)

BLUE = (50, 50, 255)

YELLOW = (255, 255, 0)

GREEN = (50, 255, 50)

RED = (255, 0, 0)

Loot types:

LOOT\_TYPES = ["weapon", "bullets", "paramedics", "armour"]

Weapon types:

WEAPON\_TYPES = ["glock", "ak47", "shotgun"]

Bullet types:

BULLET\_TYPES = ["pistols", "rifles", "shotguns"]

Armour types:

ARMOUR\_TYPES = ["heavy", "medium", "light"]

Medicine kit types:

PARAMEDIC\_TYPES = ["heavy", "medium", "light"]

Screen size:

size = (1000, 1000)

Data design:

1. Base class for all both enemies and player
2. Class for a user’s player
3. Class for an enemy

**class diagram for players and enemies:**

class People

class Player

class Enemy

**Class diagram for loot:**

class Loot

class Paramedic

class Weapon

class BulletsLoot

class Armour

Base class for player and enemies

|  |
| --- |
| **class People** |
| **Attributes:**  *- width*  *- height*  *- health*  *- speed*  *- color*  *- bricks*  *- bullets\_list*  *- health\_bar* |
| **Methods:**   * updatePlayerPosition() * getXPosition() * getYPosition() * isCollision() * move() * setSpeed() * shoot() |

**Algorithms:**

1. updatePlayerPosition(x, y)

Updates the player coordinates x and y. The arguments are new x and y coordinates.

1. getXPosition()

Method to get the x coordinate of the player.

1. getYPosition()

Method for getting the y coordinate of the player.

1. isCollision()

Check if the player has collided with any wall bricks and restrain its movement in that direction. The algorithm uses pygame.sprite.spritecollide function to determine the collisions with walls. Then it checks the direction of the brick compared to the player or enemy (up, down, left or right). The method returns the list of restraint directions (e.g. [“up”, “”, “”, “”] means there is a wall on top of the player)

1. move()

pass the method to children classes.

1. setSpeed(newSpeed)

This method sets the speed for the player: speed is equal to newSpeed.

1. shoot()

This method creates a bullet and releases it. Then, it adds the bullet sprite to the bullet sprite list (bullets\_list)

|  |
| --- |
| **class Player** |
| **Attributes:**  *- width*  *- height*  *- health*  *- speed*  *- color*  *- playerX*  *- player*  *- weight\_capacity*  *- inventory*  *- selectedWeapon*  *- bullets*  *- weapons*  *- max\_amount\_weapons*  *- loot\_group*  *- health\_bar* |
| **Methods:**   * updatePlayerPosition() * getXPosition() * getYPosition() * isCollision() * move() * setSpeed() * shoot() * getInventoryWeight() * setSelectedWeapon() * heal() * getWeaponsList() * getBulletsList() * checkLootCollision() * getInventory() * getMedicineKitsAmount() * isBulletCollisionWithEnemy() * isHitByEnemy() |

**Algorithms:**

1. getInventoryWeight()

Method for calculating the weight of items in the inventory. Loops through all the items in the inventory and calculates the sum of their weights. and returns it.

New algorithms can be implemented: when new item is added, the current inventory weight will be incremented by the item’s weight. The attribute of player class will store the current inventory weight.

1. setSelectedWeapon(value)

Method that sets the selected weapon for player accordin to the key the player pressed. If val is less than number of all weapons, than selectedWeapon = val-1.

1. heal()

If the user inputted T, Y, or U the user can use 3 types of medicine kits: large, medium and small. Different kits have different weight and healing effect. So, the largest kit has the biggest healing effect and it is the largest one of all 3 types of medicine kits. The algorithm then checks if the player has corresponding medicine kits in his inventory. If yes, it will use it to heal the player (add health point to player) and delete from the medicine kit from inventory. Also, this method validates the health points, so the health level can’t rise above 100 points.

1. getWeaponsList()

Returns the list of weapons the player possess.

1. getBulletsList()

Method that returns the list of player’s bullets in the inventory (type and amount of bullets)

1. checkLootCollision()

The method that checks if the user has collide with the loot boxes. It uses pygame.sprite.spritecollide function to get all the collisions. Then it checks the type of the loot; if the loot type is weapon, the weapon is added to the weapons list. If the loot type is bullets, then the bullets\_list is updated.

1. getInventory()

Returns the player’s inventory (list)

1. getMedicineKitsAmount()

Returns the list of the medicine kits the player posseses.

1. isHitByEnemy(enemies)

The method that checks if any of the enemies collided with the player. If it is true the enemy is killed and the player is damaged by 10 points. The score is also incremented by 10 points.

1. isBulletCollisionWithEnemy(enemies, incrementKills, incrementScore)

The method which checks if any of the player’s bullets have hit the enemis. If the bullet has hit the enemy, the enemy looses health points and the bullet is removed from the screen (removed from the bullets\_list as well). The kills counter and score counter are incremented by certain amount.

1. shoot()

The method which allows the player to shoot particular types of bullets which correspond to weapons possessed by the player. So, the methos checks if the bullet type matches the type of the selected weapon and if the number of bullets is more than zero. If all of this is true, the bullet is released and the number of bullets is decremented.

1. move(direction)

This method allows the player to move in 4 directions on the screen: up, down, right and left. It checks if the player can move in that direction, and if yes, then it changes the corresponding coordinates by speed of the player.

1. getXPosition()

Returns player’s x coordinate

1. getYPosition()

Return the player’s y coordinate

|  |
| --- |
| **class Enemy** |
| **Attributes:**  *- fieldView*  *- isAttacking*  - attackVector |
| **Methods:**   * drawHealthBar() * isCollision() * move() * getVector() * update() |

**Algorithms:**

1. drawHealthBar()

Method for drawing the healthbar for the enemy. It calls the method of healthbar class

1. isCollision()

Method that checks if there is any collision between the enemy and a wall. If yes, it restrains enemy’s movement, so it cant move in the direction where the wall is.

|  |
| --- |
| **class Game** |
| **Attributes:**  *- numBricks*  *- brickSide*  *- kills*  *- score*  *- wave*  *- enemy\_sprites\_group*  *- all\_sprites\_group*  *- bricks\_sprites\_group*  *- loot\_sprites\_group*  *- player*  *- done*  *- inventoryList*  *- scoreBoard* |
| **Methods:**   * incrementKills() * incrementScore() * createLoot() * createOutterWalls() * start() * end() * createEnemies() * reRender() * mainLoop() |

|  |
| --- |
| **class Bullet** |
| **Attributes:**  *- name*  *- width*  - height  - speed  - image  - rect: rect.x, rect.y |
| **Methods:**   * move() * draw() * update() |

**Base loot class for all loot types:**

|  |
| --- |
| **class Loot** |
| **Attributes:**  *- weight*  *- name*  - loot\_type  - width  - height  - rect: rect.x, rect.y |
| **Methods:**  *none* |

## Test Plan

Fill in the table below with at least 30 tests:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Description** | **Input** | **Expected Outcome** |  |
| 1 | Screen appears and is 800 by 600 pixels | Run the code | Screen appears on the display |  |
| 2 | Player moves right | Click D key | Player moves right 5 pixels |  |
| 3 | Player moves left | Click A key |  |  |
| 4 | Player moves up | Click W key |  |  |
| 5 | Player moves down | Click S key |  |  |
| 6 | Shooting | Left click | If the player has the weapon with appropriate bullets, he will shoot the bullet. |  |
| 7 | Looting | Player collides with loot box | If the player has enough weight capacity in his inventory, he can loot the item and the item will be added to inventory |  |
| 8 | Collision with walls | Player collides with walls | The player stops, he can’t move any farther in this direction. |  |
| 9 | Player hits the enemy with bullet | The bullet hits the enemy | If the player’s bullet hits the enemy, the enemy health is decremented. If enemy’s health <= 0, then the enemy dies (disappears) |  |
| 10 | Enemies attacking the player | The player is in enemy’s field of view | The enemy should attack the player and try to kill him. |  |
| 11 | Selecting the weapon | the user presses keys 1, 2 or 3 | If the user has a weapon in his inventory, then he selects this weapon. |  |
| 12 | The player kills the enemy | The player hits the enemy and it dies | The player’s score is incremented, the kills value is increased by 1 as well. |  |
| 13 | The player kills all enemies on the screen | No enemies in enemy group | Increase the wave value by 1. Spawn more new enemies on the screen (1 more than in last wave) |  |

# Implementation

***/\*Draft\*/***

**class People:**

***Attributes:***

*- width*

*- height*

*- health*

*- speed*

*- color*

*- playerX*

*- player*

***Methods:***

def \_\_init(self):

self.bricks = bricks

self.width = width

self.height = height

self.health = health

self.speed = speed

self.color = color

self.bullets\_list = pygame.sprite.Group()

self.image = pygame.Surface([self.width, self.height])

self.image.fill(self.color)

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

self.playerX = 0

self.playerY = 0

#health bar component

self.health\_bar = HealthBar(self.rect.x, self.rect.y, self.width\*2, self.height/3, self.health)

Plus:

* updatePlayerPosition()
* getXPosition()
* getYPosition()
* isCollision()
* move()
* setSpeed()
* shoot()
* drawHealthBar()

**class Player:**

Attributes:

* weight\_capacity
* inventory
* selectedWeapon
* bullets
* weapons
* loot\_group

Methods:

def \_\_init\_\_(self, x, y, width, height, color, speed, health, bricks, loot, inventory\_capacity):

        super().\_\_init\_\_(x, y, width, height, color, speed, health, bricks)

        pygame.sprite.Sprite.\_\_init\_\_(self)

        self.weight\_capacity = inventory\_capacity

        self.inventory = []

        self.selectedWeapon = -1

        #declare the list of the number of bullets, where 0 - pistols bullets, 1 - rifles bullet, 2 - gunshot bullets

        self.bullets = [0, 0, 0]

        #weapons[0] for glocks, 1 for ak47, 2 for shotguns

        self.weapons = [False, False, False]

        self.max\_amount\_weapons = 3

        self.loot\_group = loot

def getInventoryWeight(self):

        weight = 0

        for item in self.inventory:

            weight += item.weight

        return weight

    def setSelectedWeapon(self, val):

        if(val <= len(self.weapons)):

            self.selectedWeapon = val-1

        print(self.selectedWeapon)

def heal(self, indx):

        medicine = self.getMedicineKitsAmount()

        val = 0

        if (len(medicine[indx-1]) > 0):

            arr = medicine[indx-1]

            val = arr[len(arr)-1].healing

            print(val)

            self.inventory.remove(arr[len(arr)-1])

        if (self.health + val >= 100):

            self.health = 100

        else:

            self.health += val

def getWeaponsList(self):

        return self.weapons

def getBulletsList(self):

        return self.bullets

Plus:

* checkLootCollision()
* getInventory()
* getWeightCapacity()
* getMousePosition()
* getPlayerDirection()
* getPlayerBearing()
* getMedicineKitsAmount()
* isHitByEnemy()
* isBulletCollisionWithEnemy()
* shoot()
* move()

**class Enemy:**

**Attributes:**

* attackVector
* player
* fieldView

**Methods:**

* constructor:

def \_\_init\_\_(self, x, y, width, height, color, speed, health, bricks, player):

        super().\_\_init\_\_(x, y, width, height, color, speed, health, bricks)

        pygame.sprite.Sprite.\_\_init\_\_(self)

        self.attackVector = [0, 0, 0]

        self.player = player

        self.fieldView = 400

* attack()
* def attack(self):
* if (self.attackVector[2] <= self.fieldView):
* self.move()
* move()

def move(self):

        no\_direction=self.isCollision()

        if (self.attackVector[0] == 0):

            fraction = 0

        else:

            fraction = self.attackVector[1] / self.attackVector[0]

        xSpeed = self.speed/(math.sqrt(1+pow(fraction, 2)))

        ySpeed = xSpeed\*fraction

        #print(ySpeed)

        if (self.attackVector[0] < 0):

            #left

            self.rect.x -= math.ceil(xSpeed)

        else:

            #right

            self.rect.x += math.ceil(xSpeed)

        if (self.attackVector[1] < 0):

            #down

            self.rect.y += math.ceil(ySpeed)

        else:

            #up

            self.rect.y -= (-1)\*math.floor(ySpeed)

* getVector()
* def getVector(self):
* return self.attackVector
* update()

def update(self):

        self.attackVector[0] = self.player.rect.x-self.rect.x

        self.attackVector[1] = self.rect.y-self.player.rect.y

        self.attackVector[2] = math.sqrt(pow(self.attackVector[0], 2)+pow(self.attackVector[1], 2))

        #print(self.attackVector)

        #print("playerX: "+str(self.player.rect.x)+"  enemyX: "+str(self.rect.x))

        if (self.attackVector[2] <= self.fieldView):

            self.attack()

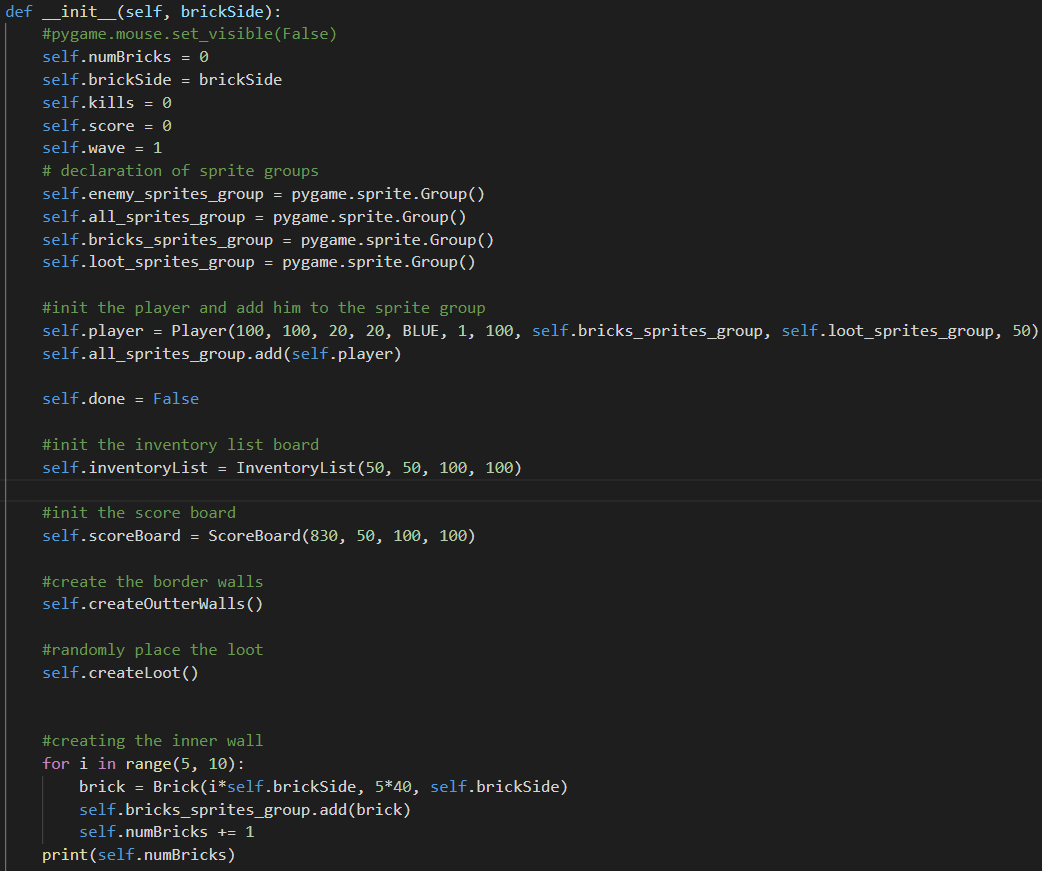
Game Class

Attributes:

* numBricks
* brickSide
* kills
* score
* wave
* enemy\_sprites\_group
* all\_sprites\_group
* bricks\_sprites\_group
* loot\_sprites\_group
* player
* done
* inventoryList
* scoreboard

Methods:

* constructor



* incrementKills()
* def incrementKills(self):
* self.kills += 1
* decrementKills()

def incrementScore(self, val):

        self.score += val

* createLoot()
* def createLoot(self):
* x = random.randint(40, 960)
* y = random.randint(40, 960)
* lootType = LOOT\_TYPES[random.randint(0, len(LOOT\_TYPES)-1)]
* if (lootType == "weapon"):
* weapon\_type = WEAPON\_TYPES[random.randint(0, len(WEAPON\_TYPES)-1)]
* loot = Weapon(x, y, 20, 20, GREEN, weapon\_type)
* print("The weapon "+weapon\_type+"was added!")
* elif(lootType == "bullets"):
* bullet\_type = BULLET\_TYPES[random.randint(0, len(BULLET\_TYPES)-1)]
* print("Bullets " + bullet\_type + " were added!")
* loot = BulletsLoot(x, y, 20, 20, GREEN, bullet\_type)
* elif(lootType == "paramedics"):
* paramedic\_type = PARAMEDIC\_TYPES[random.randint(0, len(PARAMEDIC\_TYPES)-1)]
* print("Paramedic was added!")
* loot = Paramedic(x, y, 20, 20, GREEN, paramedic\_type)
* elif(lootType == "armour"):
* armour\_type = ARMOUR\_TYPES[random.randint(0, len(ARMOUR\_TYPES)-1)]
* print("Armour "+armour\_type+" was added!")
* loot = Armour(x, y, 20, 20, GREEN, armour\_type)
* #self.all\_sprites\_group.add(loot)
* self.loot\_sprites\_group.add(loot)
* createOutterWalls()

def createOutterWalls(self):

        for row in range(0, int(1000/self.brickSide)) :

            for col in range(0, int(1000/self.brickSide)):

                if(row == 0) or (row == 1000/40-1):

                    #add block

                    brick = Brick(col\*self.brickSide, row\*self.brickSide, self.brickSide)

                    self.bricks\_sprites\_group.add(brick)

                    self.numBricks += 1

                elif(col == 0) or (col == 1000/40-1):

                    brick = Brick(col\*self.brickSide, row\*self.brickSide, self.brickSide)

                    self.bricks\_sprites\_group.add(brick)

                    self.numBricks += 1

* start()
* def start(self):
* self.done = False
* enemy = Enemy(600, 600, 20, 20, RED, 1, 100, self.bricks\_sprites\_group, self.player)
* self.enemy\_sprites\_group.add(enemy)
* self.all\_sprites\_group.add(enemy)
* self.mainLoop()
* end()

def end(self):

        self.done = True

* createEnemies()
* def createEnemies(self, quantity):
* for i in range(quantity):
* x = random.randint(40, 940)
* y = random.randint(40, 940)
* enemy = Enemy(x, y, 20, 20, RED, 1, 100, self.bricks\_sprites\_group, self.player)
* self.enemy\_sprites\_group.add(enemy)
* self.all\_sprites\_group.add(enemy)
* reRender()



* mainLoop()
* def mainLoop(self):
* while not self.done:
* screen.fill(BLACK)
* self.reRender()
* for event in pygame.event.get():
* if event.type == pygame.QUIT:
* self.end()
* if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 1):
* print("Left click!")
* self.player.shoot()
* keys = pygame.key.get\_pressed()
* if keys[pygame.K\_a]:
* #move the player to the right
* self.player.move("left")
* if keys[pygame.K\_d]:
* #move the player to the left
* self.player.move("right")
* if keys[pygame.K\_w]:
* #move the player up
* self.player.move("up")
* if keys[pygame.K\_s]:
* #move the player down
* self.player.move("down")
* #selecting the weapon
* if keys[pygame.K\_1]:
* self.player.setSelectedWeapon(1)
* if keys[pygame.K\_2]:
* self.player.setSelectedWeapon(2)
* if keys[pygame.K\_3]:
* self.player.setSelectedWeapon(3)
* if keys[pygame.K\_t]:
* self.player.heal(1)
* if keys[pygame.K\_y]:
* self.player.heal(2)
* if keys[pygame.K\_u]:
* self.player.heal(3)
* if keys[pygame.K\_LSHIFT]:
* #move the player down
* self.player.setSpeed(10)
* else:
* self.player.setSpeed(1)
* clock.tick(240)
* #EndWhile

## Functionality

Describe the game loop and any global variables or constants

For each method:

* Explain what it does and how which requirement it fulfils
* Write out pseudo code
* Draw a flowchart

game loop:

* rerender the scene
* listen for events (key press or mouse button click)

## Iteration 1

### Requirements being developed

### Errors

### Conclusion

## Iteration 2

### Requirements being developed

### Errors

### Conclusion

## Iteration 3

### Requirements being developed

### Errors

### Conclusion

# Testing

# Evaluation