James D'Arpino
CMPT_220L_111

11 December 2017

Final Write up

Abstract:

For my final project I decided to create a survival game called Project MC, however it is not the typical survival game where a player would be thrown on an island and have to survive against the horrors of that island. Instead Project MC is a survival game where the player must survive rounds out enemies that keep spawning. The goal of the player is to survive, every second the player is alive he/she will earn points. The goal is to earn the most points and earn bragging rights. The player is represented as a white box, he/she can use W, A, S, and D to move. W moves the player up, A moves the player to the left, S moves the player down, and D moves the player to the right.

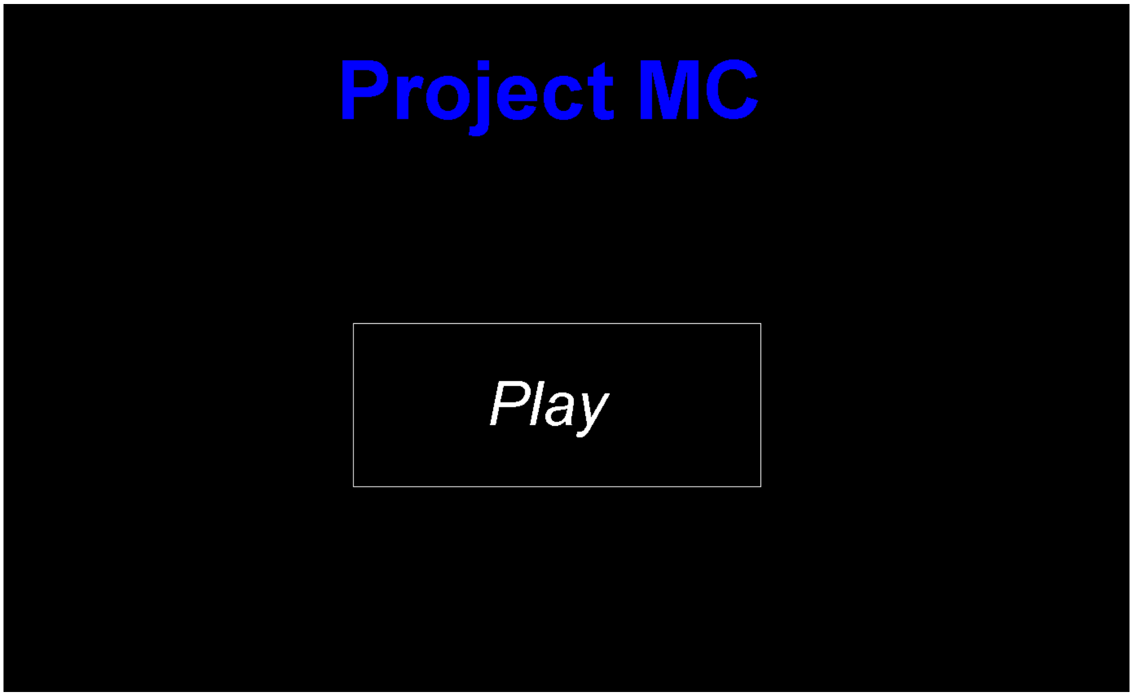
Introduction:

When I was younger I would always play java games on a website called miniclip.com. I always went on that website and would mindlessly play games for hours with no knowledge of how the games were created. I knew I wanted to learn how games were made, my main goal coming into Marist was to become a programmer that focused on video games. This is the reason I chose to make my project a video games. It was hard to come up with because a lot of code for the game was not talked about during the year, as we focused more on problems rather than games. I am very pleased with the outcome of this project. I was able to create a game, something I have always wanted to do, but have never done. The memories of working on

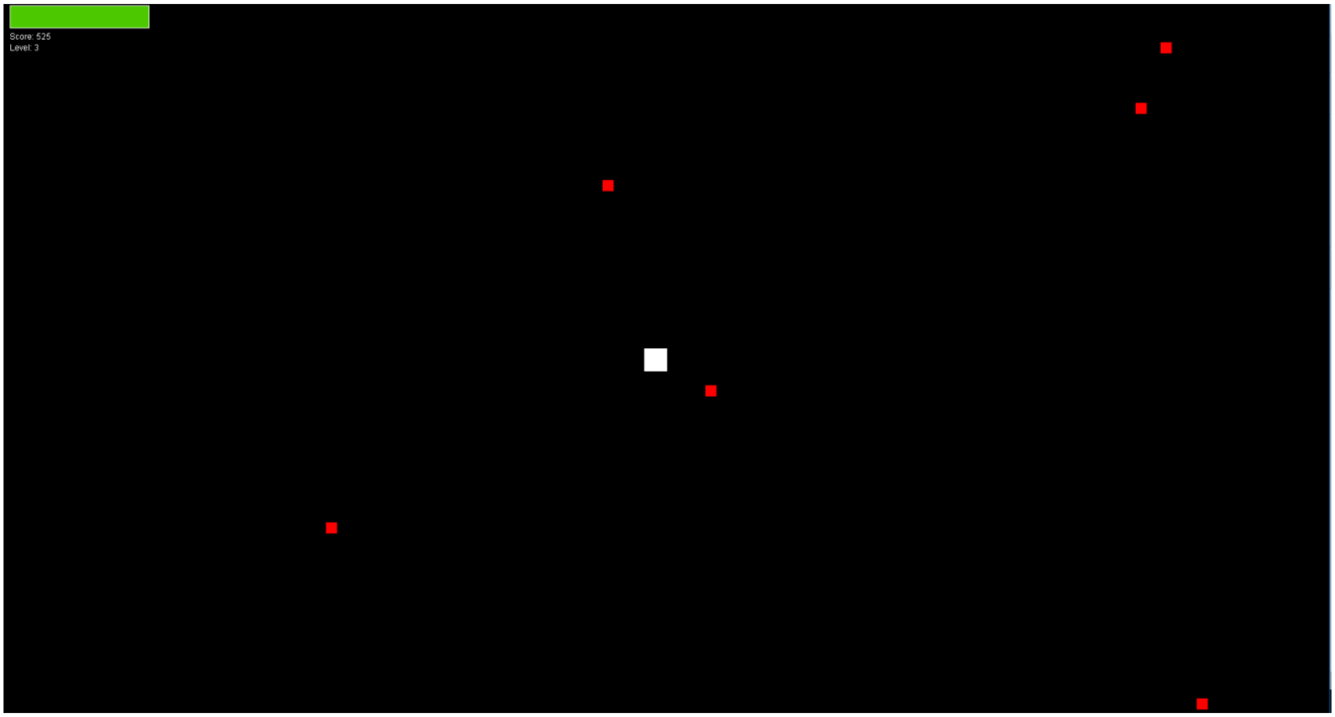
Project MC will stay with me in the future. I plan on covering a detailed system description, requirements, and a user manual for my survival game called Project MC.

Detailed System Description:

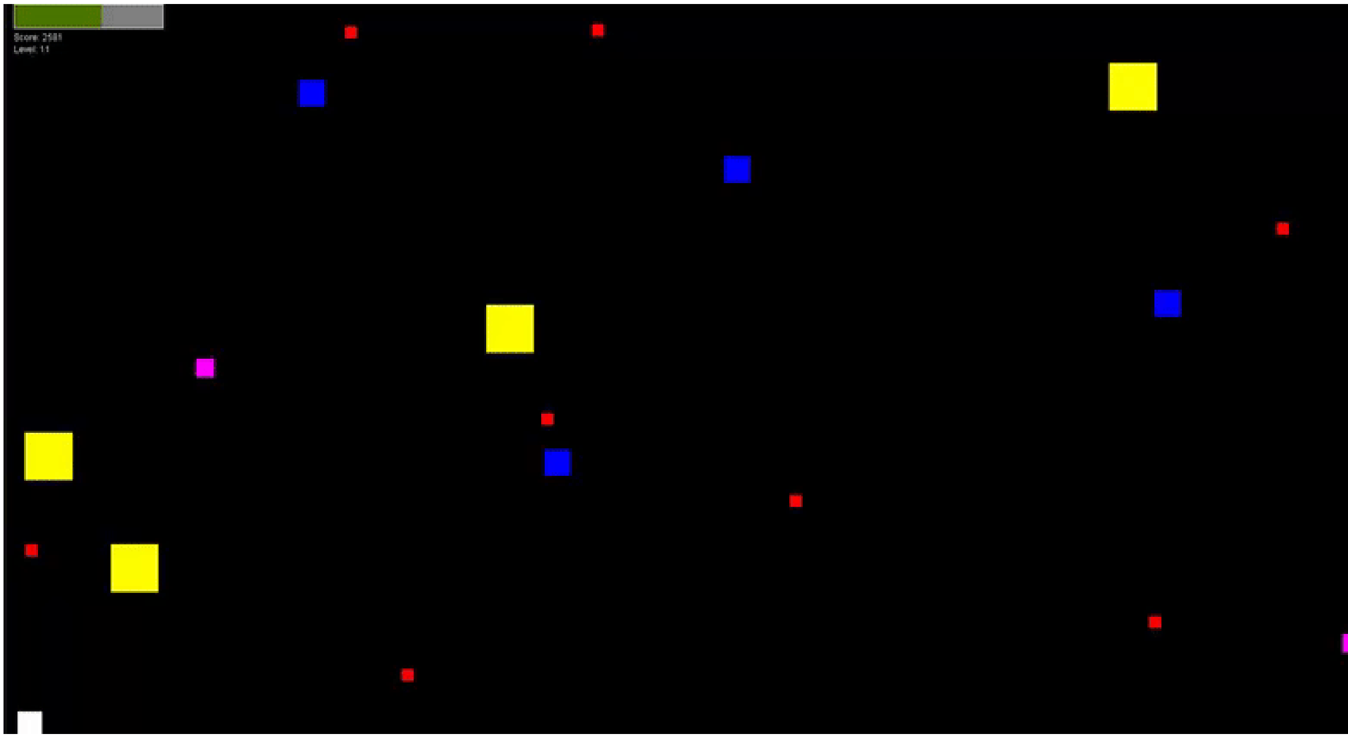
Project MC is a basic java game, used as a way to pass time, if a person wants to take a break from hours of studying he or she can play Project MC. The user just needs to start up the game and a start menu should appear.



The user then moves their cursor over to the play box and presses the left mouse button to start. The player will appear as a white box and red enemies will start spawning. The goal for the player is to avoid all the enemies, if the enemies collide with the player, the player takes damage.



As the player survives, different types of enemies will spawn, each being unique in its own way.



The red enemies are the basic ones, they tend to move slow and deal two damage. The blue enemies are advanced enemies, they move faster than the red enemies and deal four damage upon collision and spawn during the fourth level. The purple enemies are referred to as super enemies, they are faster than the advanced enemies, but tend to be smaller. These super enemies deal six damage to the player and spawn in at level six. The last type of enemy is considered the boss enemy. These boss enemies deal twenty damage to the player and spawn at level eight. These enemies are designed to kill the player upon impact, however they are much slower than any enemy to make up for their high damage output. When the player's health reaches zero, a game over menu should appear. The goal of Project MC is to earn the most points, while getting to the highest level, thus earning bragging rights among friends.

The code for Project MC is broken up into fourteen classes: Game, GameObject, Handler, Spawner, KeyInput, Window, Player, HUD, Menu, SuperEnemy, BossEnemy, BasicEnemy, AdvancedEnemy and GameOver. The classes are all separate, but work together in order to run Project MC. The main class is Game, this class calls upon methods from other classes in order to run the game.

The Game class is composed of run(), tick(), render(), clamp(int, var, int min, int max. Run() is a game loop that calculates the amount of ticks and fps for the game, this loops is an essential to running the game. Tick() and render() work together, they both update the game as the user is playing. Clamp(int var, int min, int max) makes it so players and enemies cannot go beyond the game window. The Game class also sets the size and color of the window in which the game is being run on, as well as mouse and key listeners.

The GameObject class sets the id, velocity, and position of both the player and the enemies. The Handler class is used to make edits to the objects on screen by using addObject(GameObject, object) and removeObject(GameObject object), thus allowing the ability for enemies to spawn in conjunction with the spawner class. The spawner class uses an if else statement to proceed to the next level after the score increases to 250. There are then if else statements to determine how many of which type of enemy spawns at each level.

The KeyInput class is made up of keyPressed(keyEvent e) and keyReleased (keyEvent e). KeyPressed allows for W to move the user up, A to move the user left, S to move the user down, and D to move the user to the right, keyReleased is a little different. KeyReleased makes it so that when the player releases his or her finger from any of the W, A, S, or D keys, white box they are playing as stops and doesn't continue moving.

The Window class is where the JFrame is called, the JFrame is what creates the window that is displayed after the code is running. The Player class creates the white player box used, the class also creates a border around the player that if there is a collision with another enemy then damage is dealt to the player. This also determines which enemy collides with the player and deducts damage accordingly. The HUD class creates the overlay seen when the game is booted. This is where the green health bar, the score and the levels are displayed.

The Menu class brings up the menu screen and has deals with mousePressed and mouseOver. MouseOver work together and allow the game to start if the cursor is within the play box and the user presses the left click on their mouse or trackpad. The next four classes create the enemies, these classes are called the SuperEnemy, BossEnemy, BasicEnemy, AdvancedEnemy classes. The difference with these class are only in the speed at which the enemies move, the size of the enemy, and the color of the enemy. The GameOver class is similar to the Menu class. The GameOver class creates a game over screen after the player dies.

Requirements/Literature Survey

Video games are designed for entertainment. Project MC does not stride far from the overall goal of entertainment that other games try to achieve. Project MC is a more casual game that does not require hours of play time a day. Instead it is a game where the user can play when they are bored, taking a break from studying, waiting for a movie to start, etc. Many games have done what Project MC is trying to do. The goal for Project MC to land itself as a casual game that people use for a way to pass time and try to beat their friends high scores.

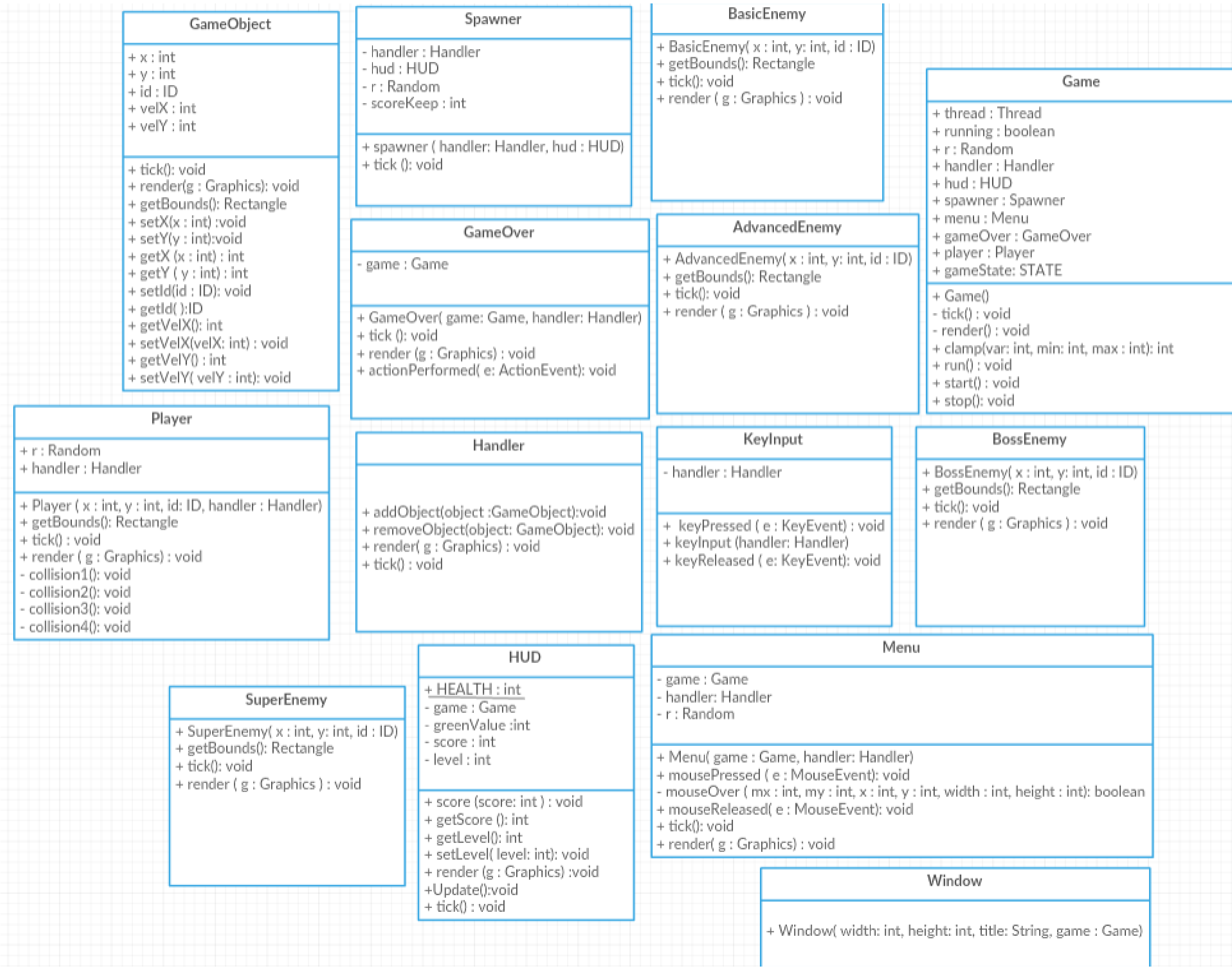
User Manual:

1) Press W, A, S, D to move.
2) Do not get hit by the other boxes
3) The bigger the box the more damage
4) Red = basic enemy, smallest amount of damage
5) Blue = advanced enemy, medium amount of damage
6) Purple = super enemy, medium amount of damage
7) Yellow = boss enemy, large amount of damage
9) Survive through level of enemies and earn points
10) Earn the best score and prove to your friends you can survive longer

Conclusion:

The goal for this project was to use the understanding of java that I have gathered through this class and use it to build a basic game. Project MC can be expanded on over time. I could add a boss at level fifteen. This would delete all enemies on screen and spawn a deadly boss that the player would need to survive for a certain amount of time. Then the game would continue as normal, with enemies spawning every 250 points earned by the player. The amount of content that could once learning more about java is insane. Project MC is not done being worked on. I will continue to work and expand on this basic idea of Project MC and hopefully turn it into a more fun and addictive game.

UML Diagrams:


GameObject
+ x : int
+ y : int
+ id : ID
+ velX : int
+ velY : int
+ tick(): void
+ render(g : Graphics): void
+ getBounds(): Rectangle
+ setX(x : int) :void
+ setY(y : int):void
+ getX (x : int) : int
+ getY (y : int) : int
+ setId(id : ID): void
+ getId( ):ID
+ getVelX(): int
+ setVelX(velX: int) : void
+ getVelY() : int
+ setVelY(velY : int): void
Spawner
- handler : Handler
- hud : HUD
- r : Random
- scoreKeep : int
+ spawner (handler: Handler, hud : HUD)
+ tick (): void
BasicEnemy
+ BasicEnemy(x : int, y: int, id : ID)
+ getBounds(): Rectangle
+ tick(): void
+ render (g : Graphics) : void
Game
+ thread : Thread
+ running : boolean
+ r : Random
+ handler : Handler
+ hud : HUD
+ spawner : Spawner
+ menu : Menu
+ gameOver : GameOver
+ player : Player
+ gameState: STATE
+ Game()
- tick() : void
- render() : void
+ clamp(var: int, min: int, max : int): int
+ run() : void
+ start() : void
+ stop(): void
GameOver
- game : Game
+ GameOver(game: Game, handler: Handler)
+ tick (): void
+ render (g : Graphics) : void
+ actionPerformed(e: ActionEvent): void
AdvancedEnemy
+ AdvancedEnemy(x : int, y: int, id : ID)
+ getBounds(): Rectangle
+ tick(): void
+ render (g : Graphics) : void
Player
+ r : Random
+ handler : Handler
+ Player (x : int, y : int, id: ID, handler : Handler)
+ getBounds(): Rectangle
+ tick() : void
+ render (g : Graphics) : void
- collision1(): void
- collision2(): void
- collision3(): void
- collision4(): void
Handler
+ addObject(object :GameObject):void
+ removeObject(object: GameObject): void
+ render(g : Graphics) : void
+ tick() : void
KeyInput
- handler : Handler
+ keyPressed (e : KeyEvent) : void
+ keyInput (handler: Handler)
+ keyReleased (e: KeyEvent): void
BossEnemy
+ BossEnemy(x : int, y: int, id : ID)
+ getBounds(): Rectangle
+ tick(): void
+ render (g : Graphics) : void
HUD
+ HEALTH : int
- game : Game
- greenValue :int
- score : int
- level : int
+ score (score: int) : void
+ getScore (): int
+ getLevel(): int
+ setLevel(level: int): void
+ render (g : Graphics) :void
+Update():void
+ tick() : void
Menu
- game : Game
- handler: Handler
- r : Random
+ Menu(game : Game, handler: Handler)
+ mousePressed (e : MouseEvent): void
- mouseOver (mx : int, my : int, x : int, y : int, width : int, height : int): boolean
+ mouseReleased(e : MouseEvent): void
+ tick(): void
+ render(g : Graphics) : void
SuperEnemy
+ SuperEnemy(x : int, y: int, id : ID)
+ getBounds(): Rectangle
+ tick(): void
+ render (g : Graphics) : void
Window
+ Window(width: int, height: int, title: String, game : Game)

References:

https://www.youtube.com/watch?v=1gir2R7G9ws&t=153s

https://www.youtube.com/watch?v=0T1U0kbu1Sk

https://www.youtube.com/watch?v=bWHYjLJZswQ

https://www.youtube.com/watch?v=5ufOPX8N1Rg

https://www.youtube.com/watch?v=3paMFMwVfWU&t=1197s

https://www.youtube.com/watch?v=KpzvlvxZkcE&t=1012s

https://www.youtube.com/watch?v=JrSjwQbTldg&t=117s

https://www.youtube.com/watch?v=Urg8AEIVyWA

https://www.youtube.com/watch?v=urye2D3_mss&t=1167s