CS102	**Fall 2020/21**	
Istructor:	**Uğur Güdükbay**	Project Group H
Assistant:	**Serkan Demirci**	

Criteria	TA/Grader	Instructor
Presentation		
Overall		

~ EscApe ~

INVICTA

Deniz Tuna Onguner, Hasan Ege Tunç, Ali Kaan Şahin, Kerem Tekik, Tarık Berkan Bilge

Detailed Design Report

Final Version

24 December 2021

Introduction

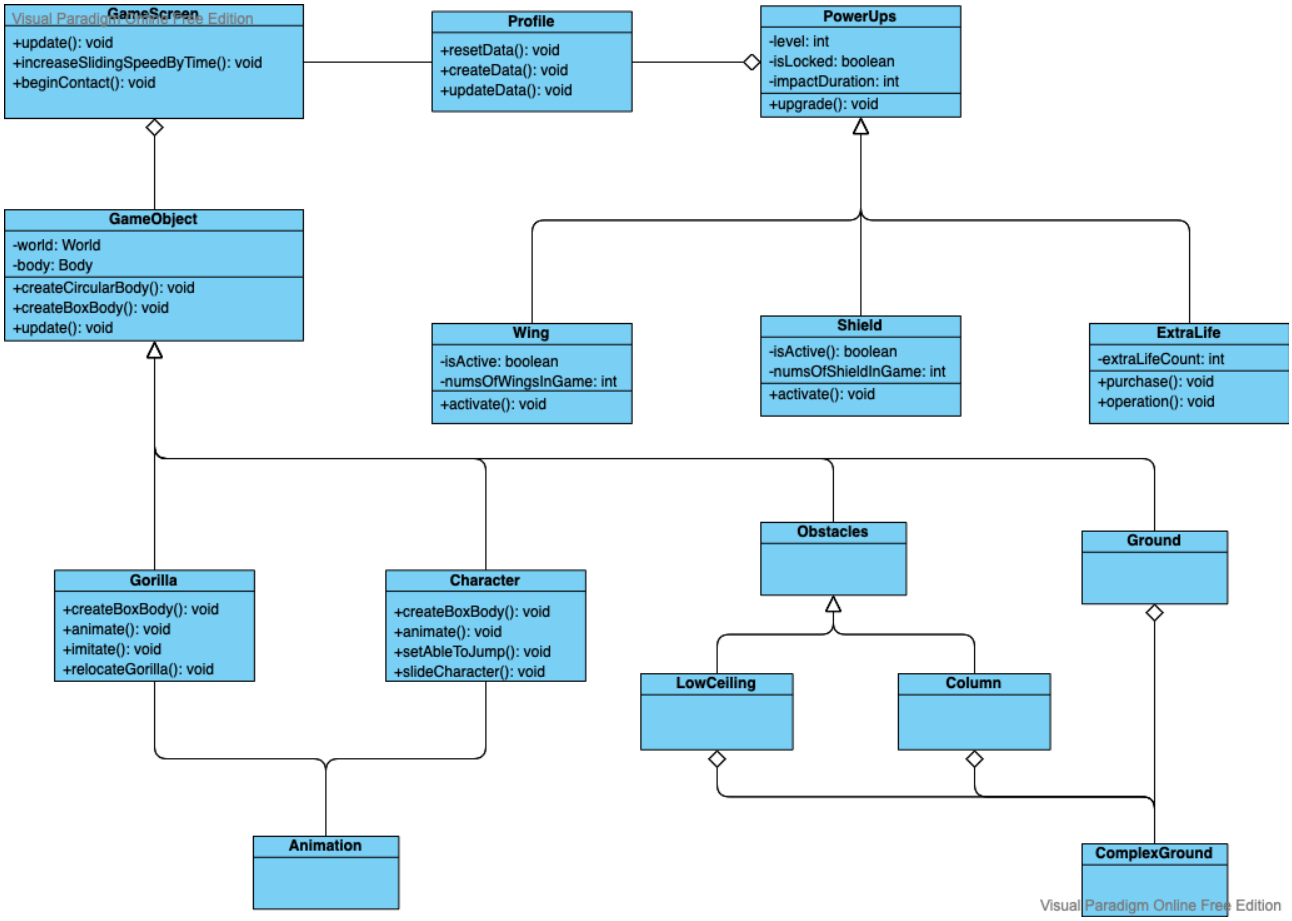
EscApe is a 2D desktop game that supports single playing. The player aims to escape a furious ape throughout the a city parkour that consists of grounds separated by spaces, and there are various obstacles on the grounds (i.e. short walls/columns, low ceilings) that the player should avoid. The player might use power-ups that are purchased from a shop menu for a better game playing.

Details

2.1.System Overview

EscApe is a desktop game that uses LibGDX library [1], and LibGDX's preferences [2] class for database.

2.2.UML Diagram



2.3.Explanation of UML Diagram

class GameScreen:

void update(): Updates background, repositions the character and sets the velocity of the character due to its position.

void moveGameObject( ): moves game object

void mapGenerator( ): generates map by creating 100 random complex grounds.

void distributeCoins( ): distribute coins calculating the position of each ground component of complex grounds

void builtColumnOrCeiling( ): builds column or ceiling randomly. While doing this, it uses an algorithm so that there will not be any obstacle that character cannot pass.

void distributeColumnAndCeilings( ): by using builtColumnOrCeiling, it positions obstacles so that they will be right above the grounds, not be hanging on air. It also decides how many obstacles will be on each complex ground by checking the number of grounds included in.

void increaseSlidingSpeedByTime(): Increases the speed of sliding background 0.5f in a certain period.

void beginContact(): Checks the contacts between character and obstacles. Implements the shield method if it is activated.

class Profile:

void resetData(): Cleaning all data, basically resets the profile progress.

void createData(): Create a new database file.

void updateData(): Saving all changes and progress into data.

abstract class PowerUps

int level: represents the level of the power-up

Boolean isLocked: shows whether the power-up is locked or not.

int impactDuration: represents the duration that power-ups are activated.

void upgrade(): Increases the level of power-ups if the profile has enough coin.

class Wing extends PowerUps

boolean isActive: shows whether wing is activated or not.

int numsOfWingsInGame: the number of wings that is given to player.

void activate(): Activates the wing for the duration time.

class Shield extends PowerUps

boolean isActive: shows whether shield is activated or not.

int numsOfShieldInGame: the number of wings that is given to player.

void activate(): Activates the shield for the duration time.

class ExtraLife extends PowerUps

int ExtraLifeCount: the number of extra life power ups of the profile.

class GameObject

world World: the place where physics system functions.

body Body: to detect collision with other game objects.

void createBoxBody(): Creates body for game objects other than coins.

void createCircularBody(): Creates body for coins

void update(): Updates the position of the bodies and moves the image to the position of body.

class Gorilla extends GameObject

void CreateBoxBody(): Creates body for the gorilla.

void animate(): Provides a more realistic view to gorilla.

void imitate(): Gorilla acts as the same with character.

void relocateGorilla(): At the beginning of the game, prevents the character to contact with the gorilla.

class Character extends GameObject

void createBoxBody(): Creates body for the character.

void animate(): Provides a more realistic view to character.

void setAbleToJump(): sets the jumping ability of character.

void slideCharacter(): makes character slide under low ceilings.

class Obstacle extends GameObject

class LowCeiling extends Obstacle

class Column extends Obstacle

class Ground extends GameObject

class ComplexGround

void moveComplexground(): moves complex ground by moving each ground element in the complex ground.

void moveToTheEnd(): moves complex ground to the end of the game map, when it passes the visible boundaries of screen.

class Animation

2.4.Task Distribution

Hasan Ege Tunç: Creation of single player game map, movement and repositioning of game objects, design of map flow, score and count increment, and detecting collisions

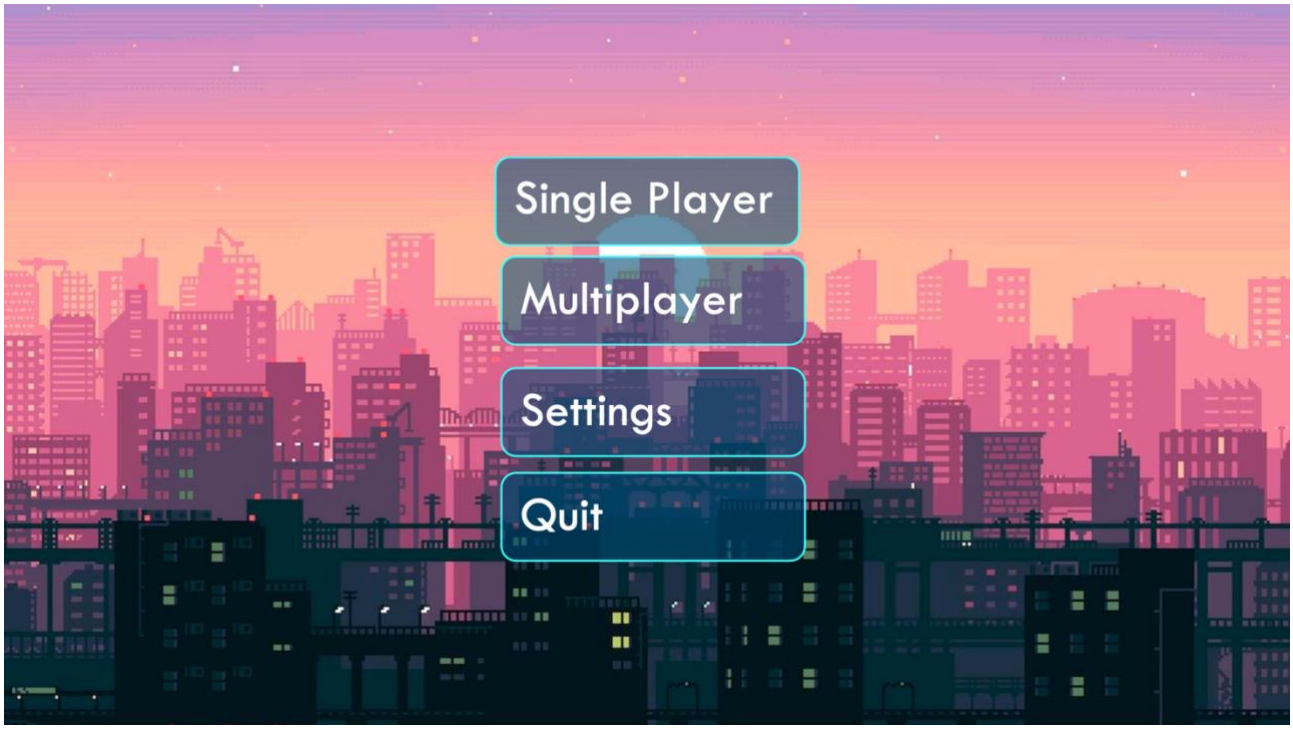
Deniz Tuna Onguner: Creation of single player game map, design of map flow, character and gorilla animations, and database

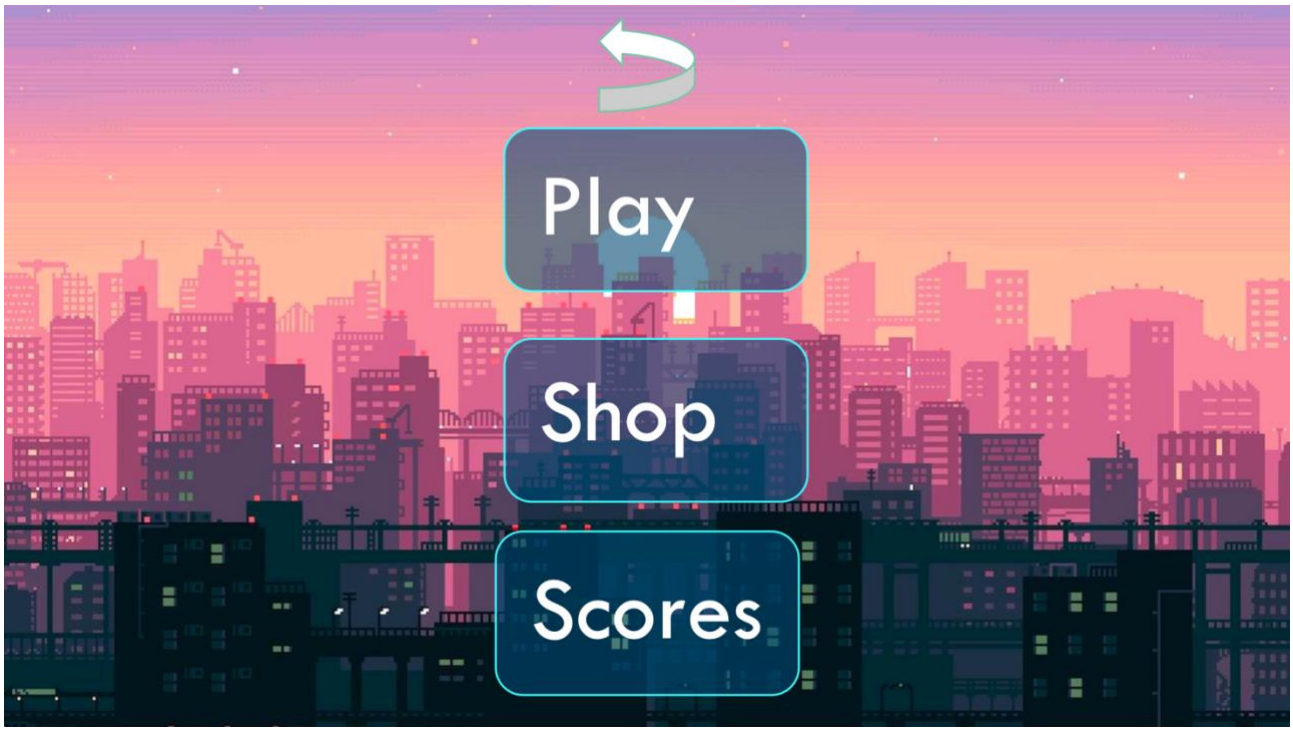
Kerem Tekik: Design and implementation of power-ups and server design in multiplayer game modes

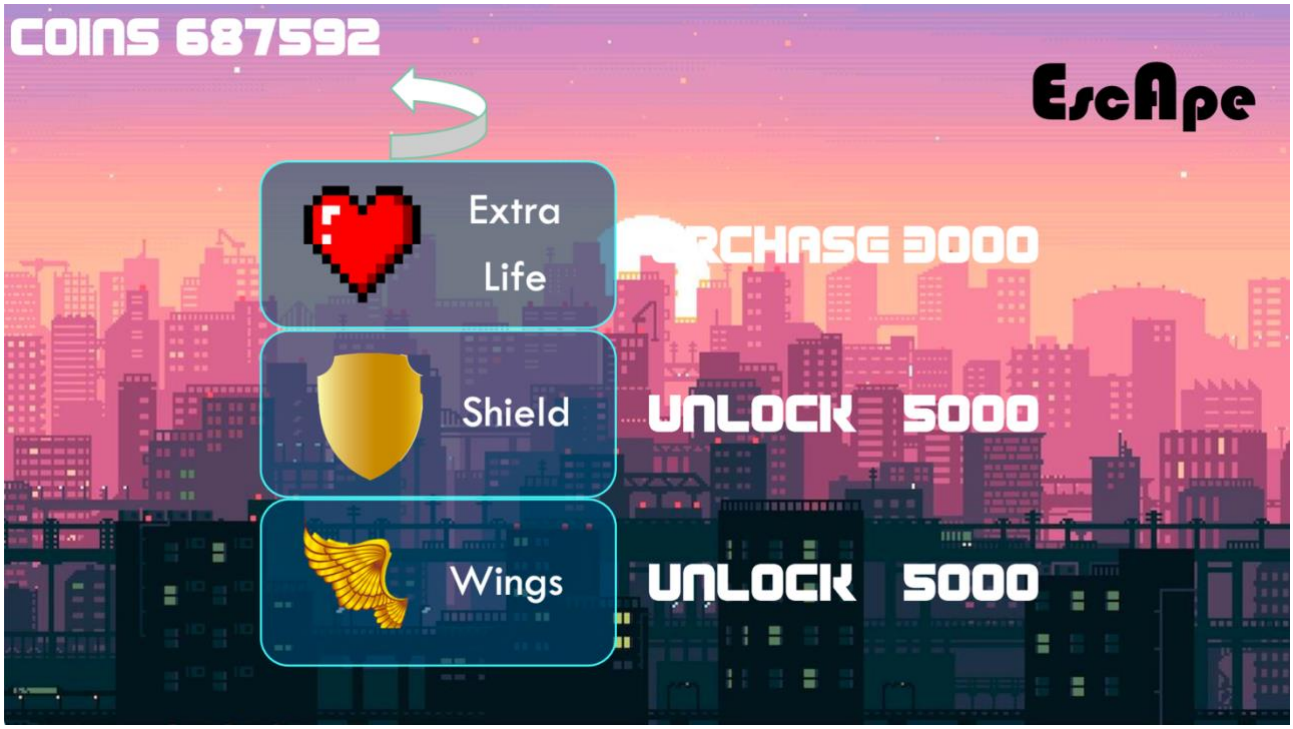
Tarık Berkan Bilge: character and gorilla design, movement of sprites, server design in multiplayer game modes

Ali Kaan Şahin: Menu design, and transition between menus, also helps multiplayer game design

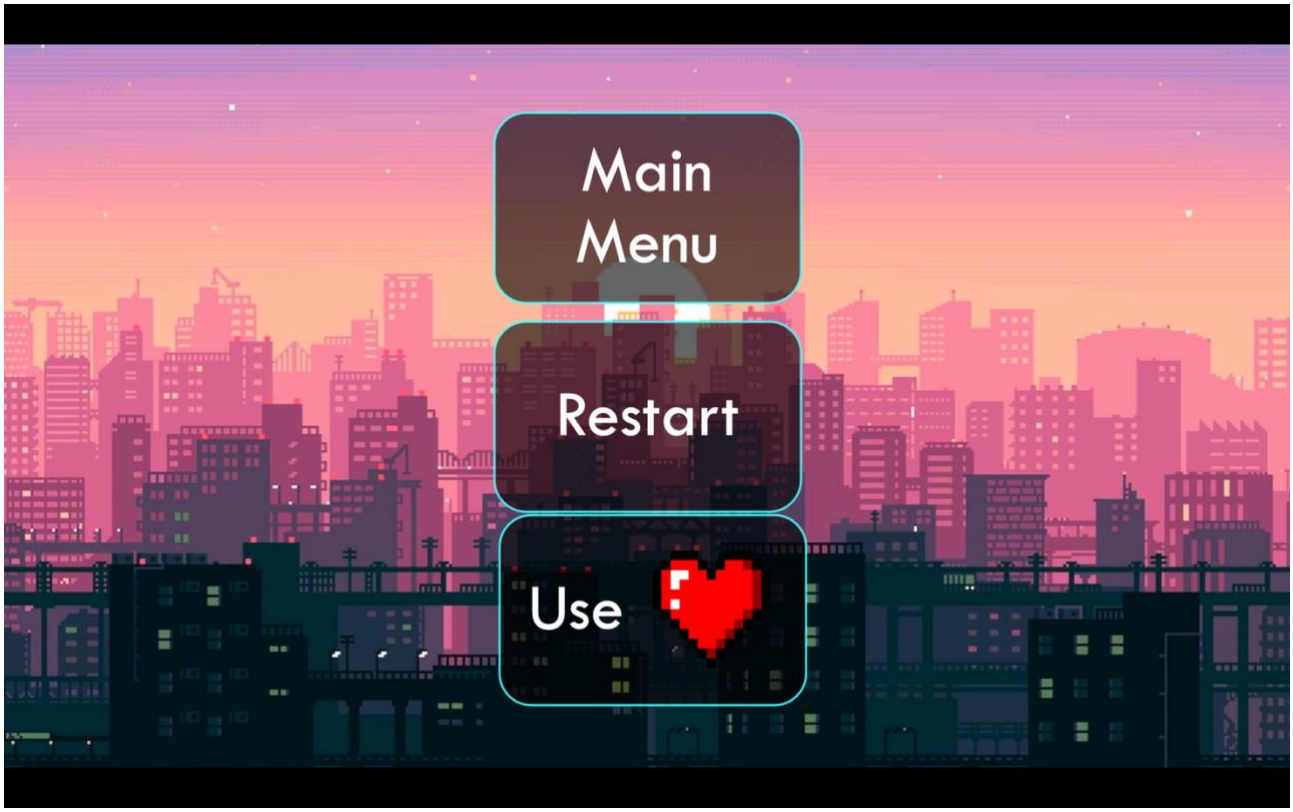
2.5. Images From EscApe






COINS 687592
EscApe
Extra
Life
Shield
Wings
UNLOCK 5000
UNLOCK 5000


POWER UPS
PASS OBSTACLES
WITH SHIELD
FLY WITH WINGS



2.6.What we could achieve and could not achieve?

What we could achieve?

- Single player
- Animated character and flowing background
- Animated gorilla
- Keeping data
- Multiple Power-ups
- Infinite Map
- Various obstacles

What we could not achieve?

- Multiplayer
- Settings menu
- Extra life power-up

Summary & Conclusions

As a freshmen group, we are proud of creating something unique, and it is valuable for us, despite the fact that we were not able to achieve all of pre-determined goals. The most difficult part of the project seems to be multiplayer game section, which we could not overcome. If we had a chance to do something different than what we did, we would start earlier to solve multi-player game issue.

To summarize who worked on which class, Hasan Ege wrote Column, Coin, ComplexGround, GameObject, Ground, Low ceiling, Obstacle classes and considerably contributed to Character, Power-ups, Gorilla, GameScreen, MainMenu and SinglePlayerMenu classes. Deniz wrote Animation, Profile, Character and Gorilla classes, he also made important contributions on GameScreen, Game Main classes. Ali Kaan wrote SinglePlayerMenu, ShopMenu, SettingsMenu, ScoresMenu, MultiplayerMenu, MainMenu, ControlsMenu, CreateServerMenu, GameOverScreen, JoinServerMenu, ProfilesMenu classes. Kerem wrote PowerUps, Shield, Wing and ExtraLife classes and had a significant share in GameScreen and GameObject classes. Tarık majorly worked on multiplayer game and his work can be seen by looking at the branched part of the project. He also played a role in gorilla and character classes.

All of group members worked on implementation part at least three-days, when report writing is added to this duration, it grows to much higher extent. Finally, what we dislike is that when we could not handle some problems like multiplayer, we felt hopeless and desperate.

REFERENCES

[1] *LibGDX*. libGDX. (2021, December 1). Retrieved December 24, 2021, from https://libgdx.com/

[2] Preferences (libgdx API). (2021, February 26). Retrieved December 24, 2021, from https://libgdx.badlogicgames.com/ci/nightlies/docs/api/com/badlogic/gdx/Preferences.html