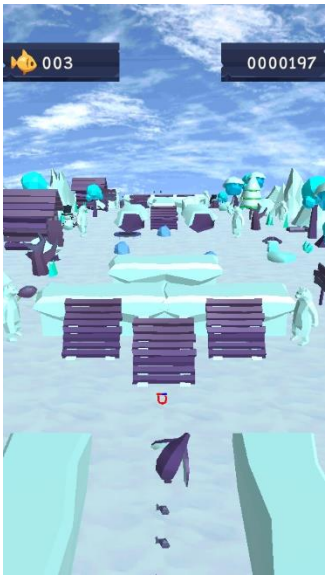
The brave Pingu

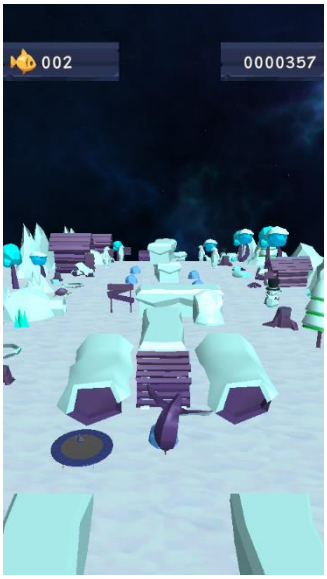
Darius David, 2021

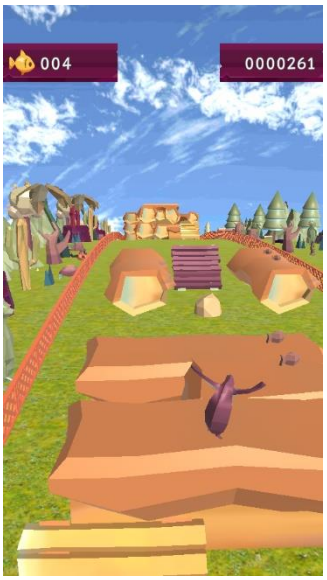
1. Abstract

This report includes an overview of my proposed mobile game, as well as comprehensive details about the app's features, target users and objectives, a comparison of similar apps and used links references.

IN-GAME SCREENSHOTS







2. Goal and users

This game is suited for people who would enjoy games that test agility, reflexes, and skills. However, this game is targeted at an audience of children, because they are more likely to play games on their phones, but everyone who wants to relax and to spend some of his free time can try this game.

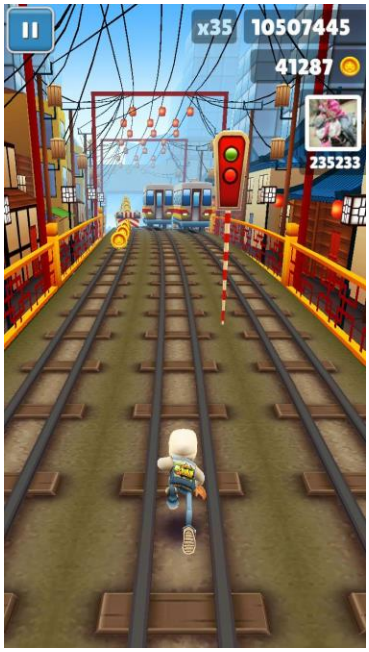
3. Introduction

The fundamental purpose of this project is to explore how to create a game with Unity3D game engine. Another aim was to get familiar with the basic processes of making a game. In this study, I will study the theoretical frameworks of game engine and mainly focused on the Unity3D game engine. Then the theoretical knowledge will be applied into practice. The final project will be an endless running game, which allows the layers getting points by keep moving on the ground and colleting some collectable items that appear during the run. In addition, the players need to dodge the enemies and pay attention to all the obstacles.

4. State of art

Other successful games in the market with similar appeal would be:

Subway Surfers:



Subway Surfers is a classic offering that is highly rated by millions of players across the world. It's simple enough with easy controls, and you'll join in the chase and jump on and off power lines and trains in the subway in a bid to escape the clutches of the Inspector.

TEMPLE RUN:

Temple Run is undoubtedly one of the most popular and downloaded games ever. There are several in the franchise to choose from, but I'm talking only about Temple Run (1) which is the first game from this series. Playing the game, you'll need power-ups, coins, and gems to speed you along your way, all the while being chased by a monster. You can use the in-game gold coins or make an in-app purchase to unlock further characters.

5. WHAT IS THE ORIGINAL CONTRIBUTION OF THE AUTHOR?

The app will include, obviously, some basic features as the games mentioned above, but it will consist also of my own contributions (some ideas for different implementations on world generation, powerups, front end, assets, animations, ···).

6. DEVELOPMENT PLAN

For the programming language, I choose C#, because it is an accurate, simple and object-oriented programming language which derived from C and C++. Also, C# inherits the powerful features of C and C++. At the same time, C# gets rid of some complex characteristics from C and C++. On the other hand, C# does not apply to write very high performance code, and lack of key features that high-performance applications required.

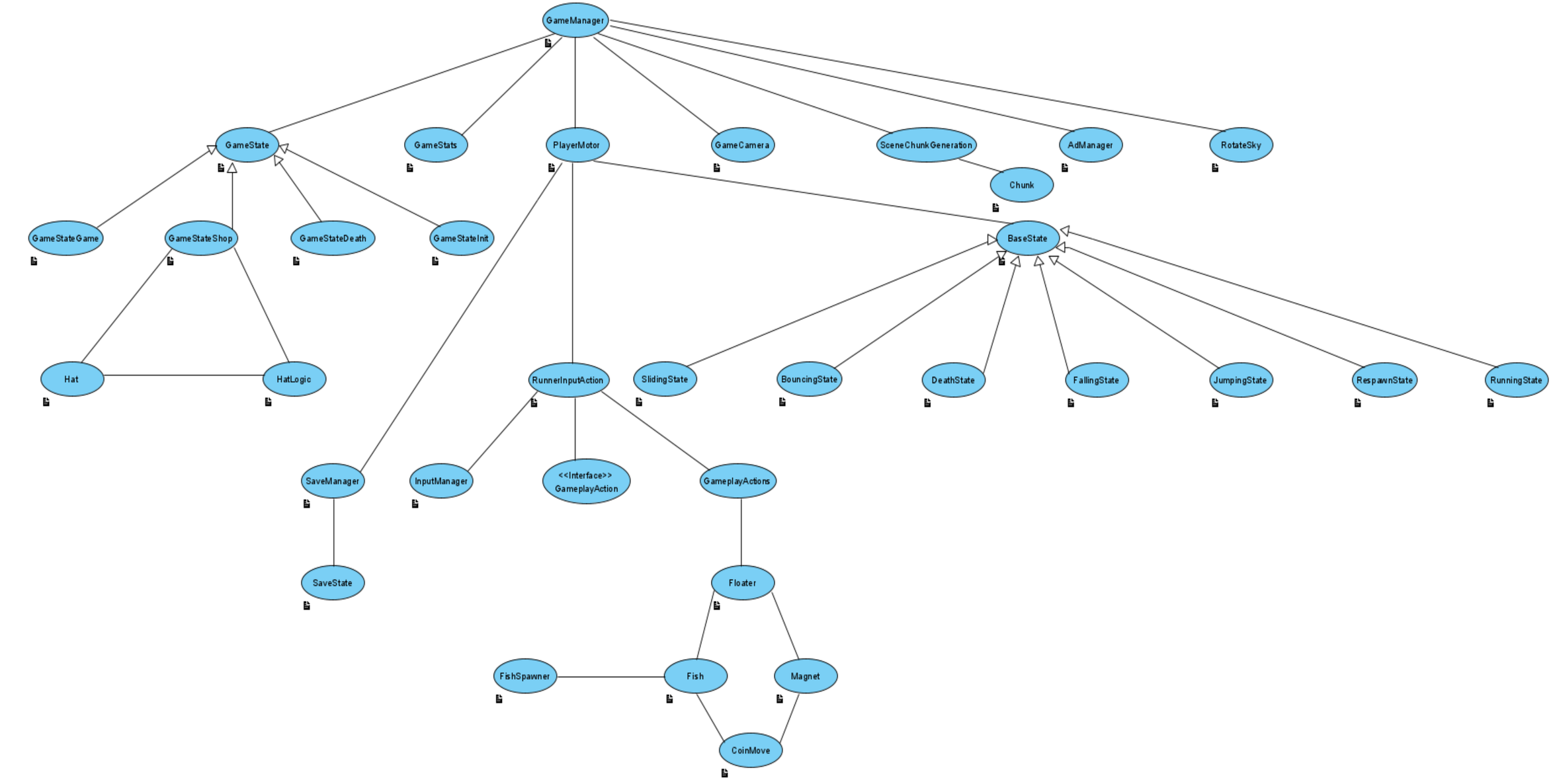
The version of Unity I choose to work with is Unity 2019.4.2f1, and another technologies I use are:

- Universal Render Pipeline
- Mechanism animator
- ShaderGraph
- Android Build
- Pooling
- Input System
- State machine
- CineMachine
- Serialized Save State Unity Advertisement

7. References

1. Unity Documentation, 2016. Unity Manual, Animation, Animation Reference, Animation Controller, Animation States. http://docs.unity3d.com/Manual/class-State.html
2. Unity Documentation, 2016. Unity Manual, Unity Graphics, Graphics Overview, Materials, Shaders & Textures. http://docs.unity3d.com/Manual/Shaders.html
3. Unity Documentation, 2016. Unity Manual, Physics, Physics Overview, Colliders. http://docs.unity3d.com/Manual/CollidersOverview.html
4. Unity Documentation, 2016. Unity Manual, Working in Unity, Creating Gameplay, GameObjects. http://docs.unity3d.com/Manual/GameObjects.html
5. Unity Documentation, 2016. Unity Manual, Working in Unity, Creating Gameplay, Scenes. http://docs.unity3d.com/Manual/CreatingScenes.html
6. Fischer, Fabian. 2014. Criteria for Strategy Game Design. http://www.gamasutra.com/blogs/FabianFischer/20141201/231243/Criteria_for_Strategy_Game_Design.php

DOMAIN MODEL



Script	Description	Important functions
GameManager	A GameManager is something that manages the game and keeps track of what state the game is in, manages the menu/pause systems, records and stores information for various purposes (audio/video settings, control bindings, game save data).	SignInToGooglePlayServices(), ChangeState(), ChangeCamera(), ChangeToSummer(), ChangeToWinter()
GameState	An abstract class which handles all the possible states the game could be in.	Construct(), Destruct(), UpdateState()
GameStateInit	The initial state of the game when you enter it.	|| - || - || + OnPlayClick(), OnShopClick(), OnAchievementClick(), OnLeaderBoardClick(), destroySummer(), destroyWinter()
GameStateGame	A state that launch the player into the game.	|| - || - || + OnCollectFish(), OnScoreChange()
GameStateDeath	A state in which the player dies.	|| - || - || + TryResumeGame(), ResumeGame(), ToMenu(), EnableRevive()
GameStateShop	A state in which the player visits the shop.	|| - || - || + OnHomeClick(), PopulateShop(), OnHatClick(), ResetCompletionCircle()
Hat	Scriptable object representing a hat.	-
HatLogic	Function that manages the behavior of hats (placing them on the player, swapping between them).	SpawnHats(), DisableAllHats(), SelectHat()
GameStats	Keeps track of player's statistics.	SendAchievementProgress(), CollectFish(), ResetSession()
RotateSky	Photo-based script which fakes the movement of the sky.	-
AdManager	A manager of all the ads in the game.	ShowRewardedAd()
Chunk	Script that handles the obstacles.	ShowChunk(), HideChunk()
GameCamera	Enumeration for all possible camera angles.	Init, Game, Shop, Respawn
PlayerMotor	The main controller of the player.	SetSpeed(), SnapToLane(), ChangeState(), ApplyGravity(), PausePlayer(), ResumePlayer(), RespawnPlayer(), ResetPlayer(), OnControllerColliderHit(), OnTriggerEnter()
SaveManager	Save and load data from the device.	Save(), Load()
SaveState	Data structure that tells the device what to save.	SaveState()
RunnerInputAction	Script generated automatically from an input system.	-
InputManager	Deals with the inputs from the user.	SetupControl(), OnStartDrag(), OnEndDrag(), OnTap()
Floater	Make the objects float up & down while spinning.	-
Fish	The in-game "coins".	OnTriggerEnter(), PickupFish()
FishSpawner	Spawn the chunks of fish.	OnEnable(), OnDisable()
Magnet	Powerup for pulling the coins.	OnTriggerEnter(), ActivateCoin(), ReduceBar()
CoinMove	Moves the coin to the player when a magnet is active.	OnTriggerEnter()
BaseState	The parent class for running, jumping, sliding, and the other states the player could be in.	ProcessMotion()
RunningState	The state in which the player runs.	|| - || - || + Transition()
JumpingState	The state in which the player jumps.	|| - || - || + Transition()
FallingState	The state in which the player falls from the sky/from a jump.	|| - || - || + Transition()
SlidingState	The state in which the player slides under the obstacles.	|| - || - || + Transition()
BouncingState	The state in which the player bounces on a trampoline.	|| - || - || + Transition()
RespawnState	The state in which the player is being respawned.	|| - || - || + Transition()
DeathState	The state in which the player dies.	|| - || - || + Transition()