By: Chen Ostrovski

Software Design Document

Introduction

This software design document describes the architecture and system design of my final project. The project is designed to compare the gameplay and design differences between five algorithms that create video game levels dynamically, using Procedural Generation. The main goal of the project is to expose the user to levels which are a product of said algorithms as opposed to manually created levels that they are already familiar with. In this document we will review the design theme of the project;
System Overview, System Architecture, Different kind of data designs and more.

Definitions and Acronyms

This section provides definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions will be used in the SDD and are most likely not known to the audience:

- RRP – Random Room Placement
- BSP – Binary Space Partitioning
- RPC – Random Point Corner
- DW – Drunkard's Walk

System Overview

At the beginning of the game the player will enter a registration screen where he will be able to enter his personal details. From there the details will be saved in AWS Dynamo DB in order for his character and achievements to be preserved every time he enters the system. If the player is already registered in the system he will be transferred directly to the main screen, where he can choose whether to start a new character and give it a name or play with one of his previous characters. In addition, the screen will also contain start and exit buttons (start or close the game). The player will be given the option of choosing the difficulty of the game.

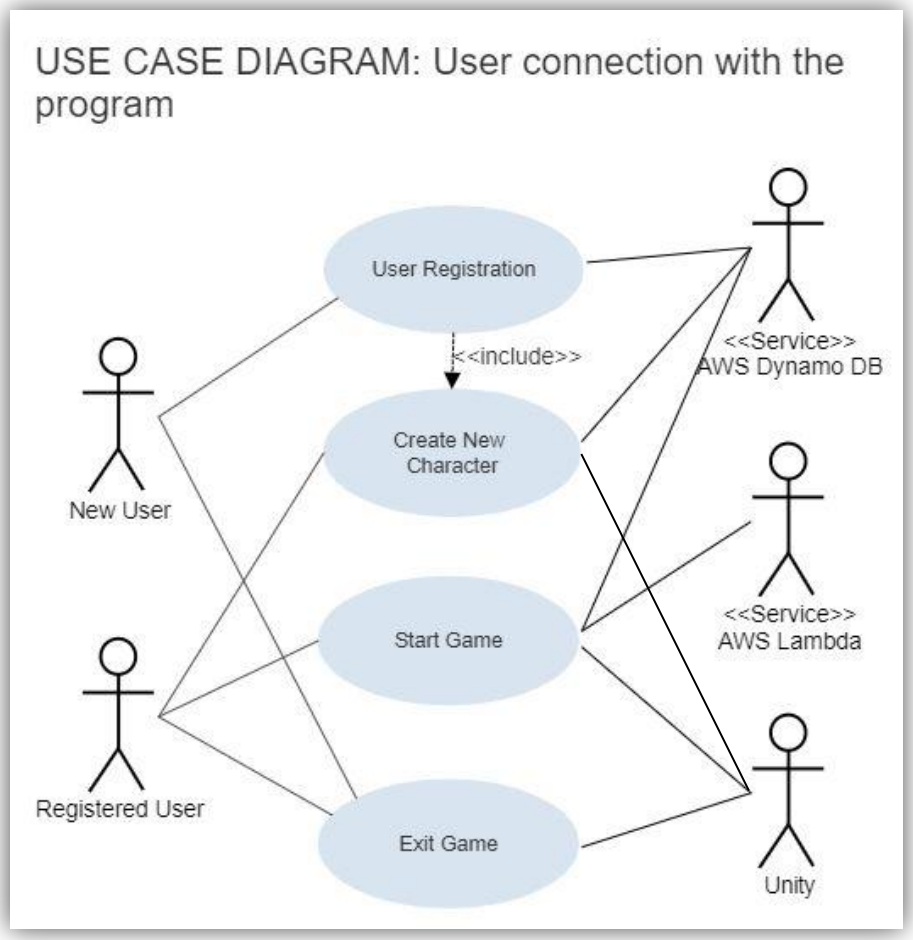
System Architecture

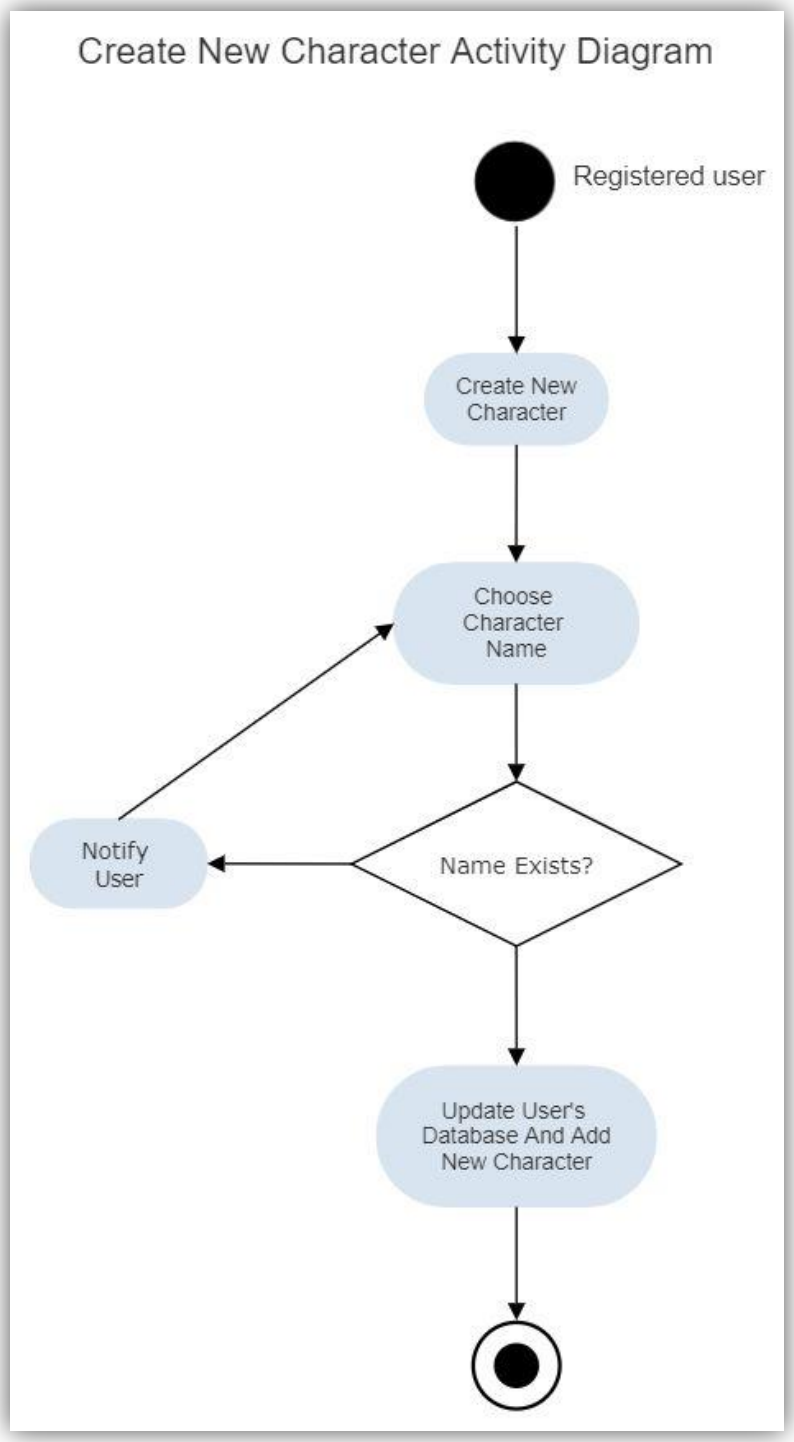
The system is partitioned and then assigned to subsystems; each subsystem has different roles and responsibilities assigned to it. The subsystems are: Unity, AWS Lambda and AWS Dynamo DB. The subsystems collaborate with each other in order to achieve the desired functionality.

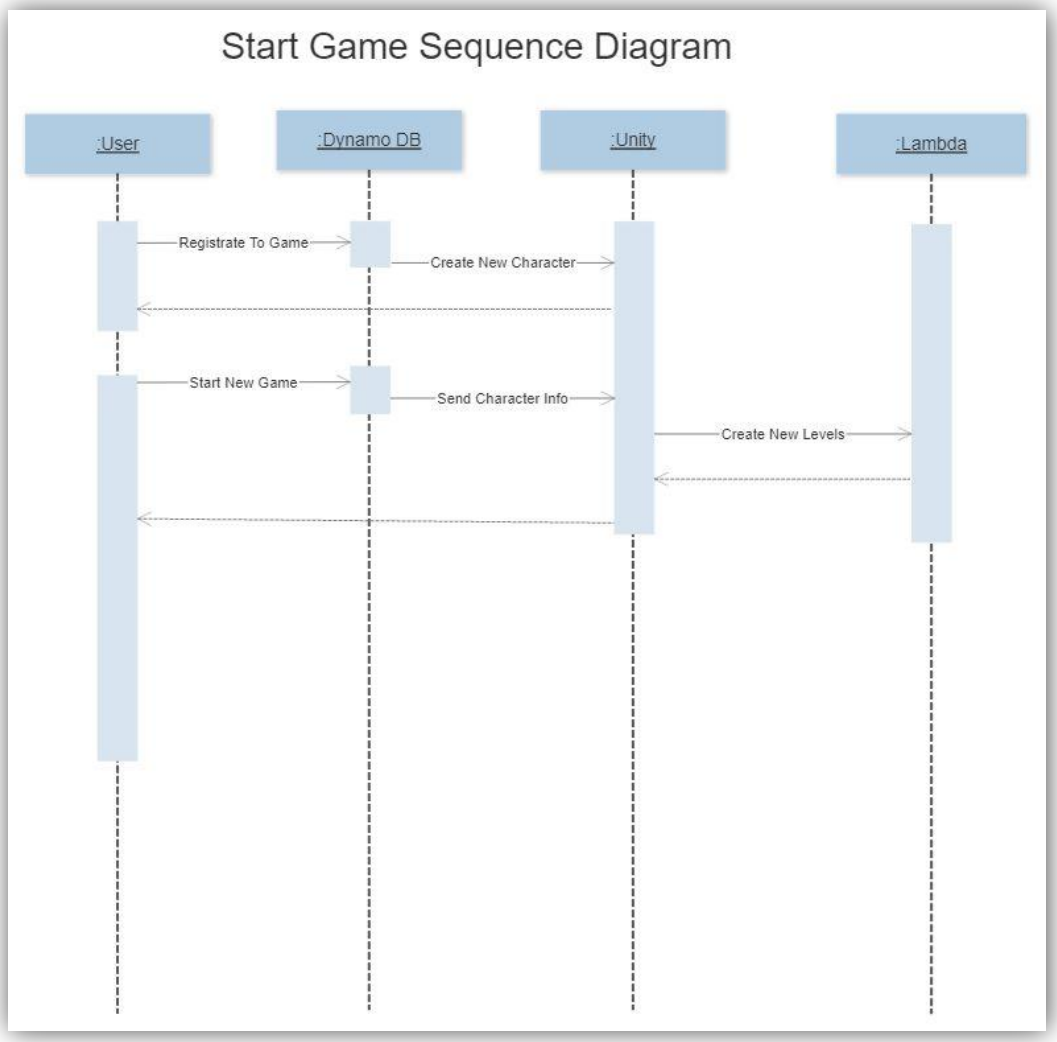
- **Unity** - With the help of this subsystem, the user can indirectly address the other subsystems. The system is mainly used to see through communication with the user. Responsible for the design of the game world and creating various objects that contribute to the user's game experience.

- **AWS Lambda** - The user does not have direct access to this system, which means that he cannot contact it or change things in it. This subsystem contains the five algorithms that create the levels of the game: RRP, BSP, RPC and DW algorithms. Each one of these creates a game level differently.

- **AWS Dynamo DB** - The user does not have direct access to this system. The system contains all the information about the user and his progress in the game. The database will contain data about the user's characters, such as: number of characters, level, life, game progress, items and more.




Create New Character Activity Diagram
Registered user
Create New Character
Choose Character Name
Name Exists?
Notify User
Update User's Database And Add New Character



Data Design

After the user is registered to the system all his info is transferred into the Dynamo DB that's provided by AWS. The user will be able to add info with tools used in Unity UI and after clicking the "Save" button, data will be transferred to storage. The database consists of the following data: Name, Password and Email. Also, during the game additional data will be added, for example, number of the player's characters and game status. Each character has its own fields such as: name, level, items and more.

Component Design

The Main Component in the game is the Player; which has many components like current data, movement, attack, talking to other characters in the game. In the given pseudo code we see an example from the player's movement script:

```
void Update()
   {
      float horizontal = Input.GetAxisRaw("Horizontal");
      float vertical = Input.GetAxisRaw("Vertical");
      Vector3 direction = new Vector3(horizontal, 0f, vertical).normalized;

      if (direction.magnitude >= 0.1f)
      {
         float targetAngle = Mathf.Atan2(direction.x, direction.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
         float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref   turnSmoothVelocity,
turnSmoothTime);
         transform.rotation = Quaternion.Euler(0f, angle, 0f);

         Vector3 moveDir = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;
         controller.Move(moveDir.normalized * speed * Time.deltaTime);
      }
   }
```

Interface Design

Unity is a cross-platform graphics engine developed by Unity Technologies and used to develop video games for computers, consoles, smartphones and websites. With the help of this engine the user will be able to download the game to their personal computer. The user will be able to use this system in order to explore all the expected features and get feedback and information.