System and Software Architecture Description (SSAD)

Doodle Dash (Preliminary Title)

Candice Academia

Johnny Chi

Nicholas Ruiz

Robert Steiminger

Stephen Sing

San Jose State University

Spring 2017

Dr. Ramin Moazeni

14 March 2017

Version History

Date	Author	Version	Changes made	Rationale
03/14/16	SS	1.0	• Original template for use with Instructional ICM-Sw v1.0	• Initial draft for use with Instructional ICM-Sw v1.0

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1. Purpose of the SSAD

The purpose of the system and software architecture document is to outline a simple plan to execute the development of our game creation. This document will list down different diagrams in the design process ranging from software capabilities to hardware restrictions. This document will also structurally define each actor as well as its hierarchy in the class diagrams.

1.2. Status of the SSAD

VERSION 1.0

This version goes off of the original template and describes basics of the software and hardware architecture. Currently the development process is still in its early cycle as we work to develop a backbone for the prototype. This version goes over the basic structural design and hardware being targeted for development.

2. System Analysis

2.1. System Analysis Overview

The final deliverable will be a simple mobile phone game. The idea of the game will be simple. All the user has to do is tap the screen in order to jump. Horizontal motion will be set, as well as the levels. The game will be more in the vein of something like Geometry Dash, compared to Flappy Birds. The difficulty of the game will come from timing the jumps in order to avoid obstacles. The rhythm of jumps will ideally be synced to music or create a rhythm. The system will have the ability to keep scores and track leaderboards as well as allow the user to customize game settings.

2.1.1. System Context

Figure 1: System Context Diagram

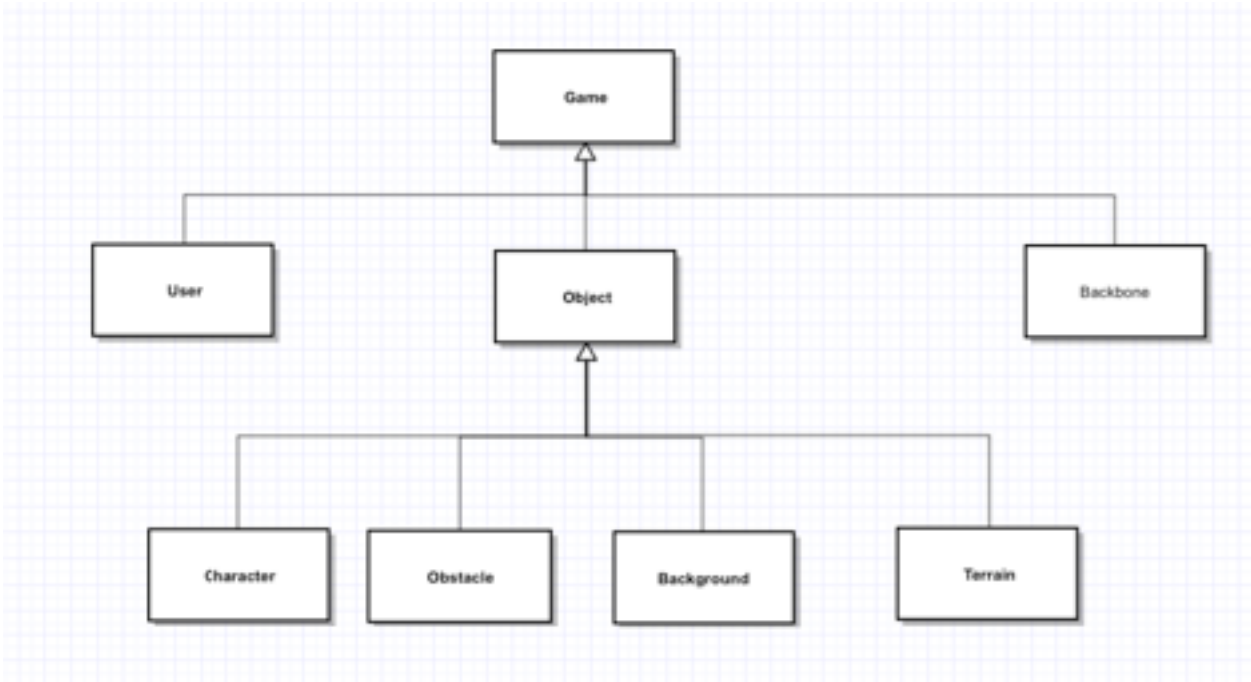
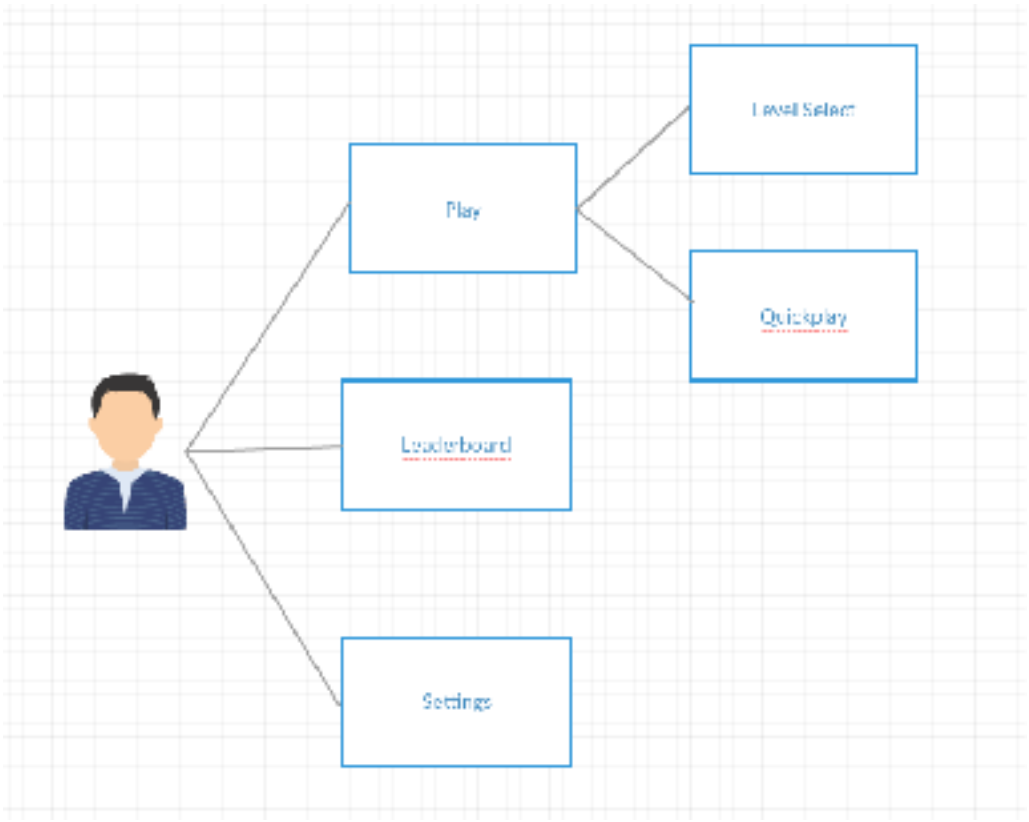


Table 1: Actors Summary

Actor	Description	Responsibilities
Object	Framework for objects in game	Responsible for detecting boundaries used to determine collision as well as specify height and width of objects
Character	The main character user will be controlling	Responsible for responding to specific inputs by user that correlate to different functions.
Obstacle	Different obstacles user will face	Describe different obstacle behaviors
Background	Serve as the background of the game	Describe the placement of the background in correlation with the character. Will also nullify collision detection
Terrain	Will serve as the platform to which obstacles and characters are on	Responsible for defining start and end goals as well as different X and Y coordinate positions and calculating whether movement is capable of reaching certain terrain
User	Serve to store user information	User will keep track of scores and preferences set by the user
Backbone	Will serve as the backbone for the game logic	Will set up the game environment as well as determine game rules and collision detection

2.1.2. Behavior

Figure 3: Process Diagram



2.1.2.1. Capability x

2.1.2.1.1. Process y

Table 3: Process Description

Identifier	Android development project
Purpose	To develop a simple game for Android
Requirements	Android phone running compatible firmware for game
Development Risks	Users may not have compatible device
Pre-conditions	Condition that the user has installed the application on their selected device
Post-conditions	None

Table 4: Typical Course of Action

Seq#	Actor's Action	System's Response
1	User presses play	System navigates to additional menu displaying level select and quickplay
2	User selects level select	System navigates to additional menu displaying all levels
3	User selects level	System generates level and User plays level
4	User ends play	System shows current score compared to high score

Table 5: Alternate Course of Action

Seq#	Actor's Action	System's Response
1	User selects level that is locked	System responds saying that the level is lock navigates back to level select page
2	User selects level that is unlocked	System generates level and User plays level
3	User ends play	System shows current score compared to high score

Table 6: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	User presses play	System navigates to additional menu displaying level select and quickplay
2	User selects quickplay	System automatically generates an unlocked level and User plays level
3	User ends play	System shows current score compared to high score

2.2. System Analysis Rationale

At this time there are no aspects of analysis that may cause high risk of confusion and misunderstanding. More information will come in later version updates of this document

3. Technology-Independent Model

3.1. Design Overview

3.1.1. System Structure

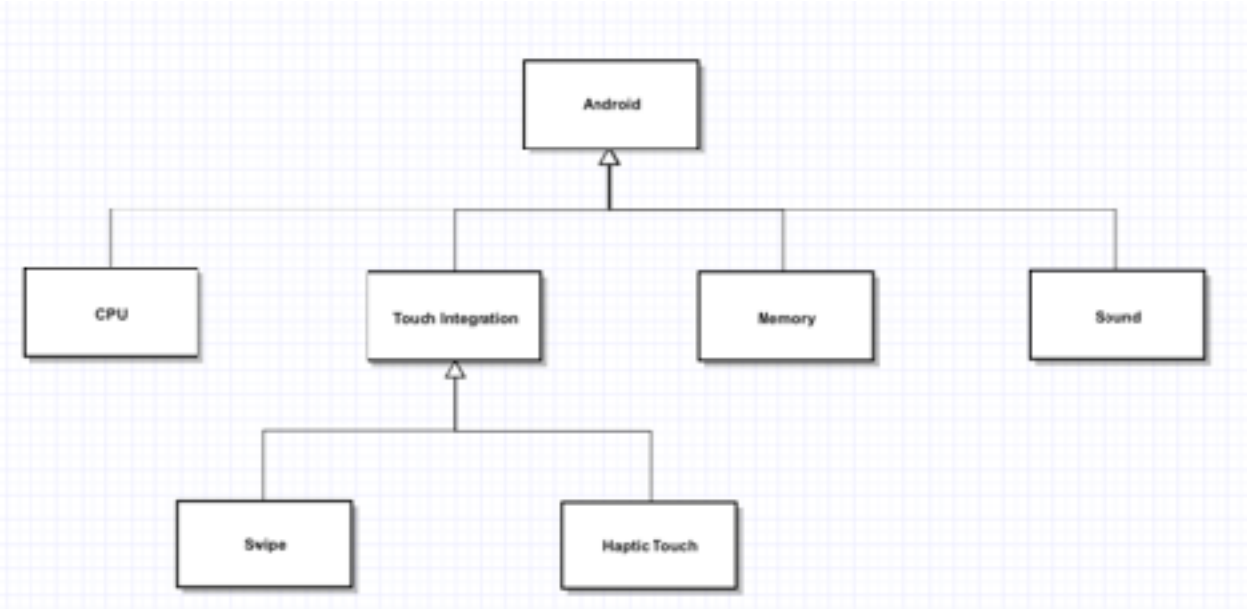


Figure 5: Hardware Component Class Diagram

Figure 6: Software Component Class Diagram

Table 7: Hardware Component Description

Hardware Component	Description
CPU	The CPU will process the game
Touch Integration	Touch integration will serve as the primary interaction tool with our users and the game
Swipe	Swipe will be used mainly to navigate through menus in game
Haptic Touch	Haptic Touch will be important as a way for the user to interact with the game. It will serve as the majority of the inputs associated with the games
Memory	Memory will be used to store the users scores as well as settings
Sound	The sound will play interactive sounds when the user interacts with specific actions in game as well as correspond to specific objects in game

Table 8: Software Component Description

Software Component	Description
Android	The game will then be ported into Android Studio for compatibility with Android devices
Unity	Unity will handle most of the work and most of the game will be developed under the Unity IDE
Photoshop/InDesign	Both these programs will be used to create and design our in game sprites and animations as well as any graphical models needed

3.2. Design Rationale

Our design rationale stemmed from traditional development techniques emphasizing the MVC model of development. We used android to meet the project specifications and wanted to specifically develop in Unity in order to learn the industry standard in game development as well as utilize its powerful engine to streamline the development process

4. Technology-Specific System Design

4.1. Design Overview

4.1.1. System Structure

Figure 12: Hardware Component Class Diagram

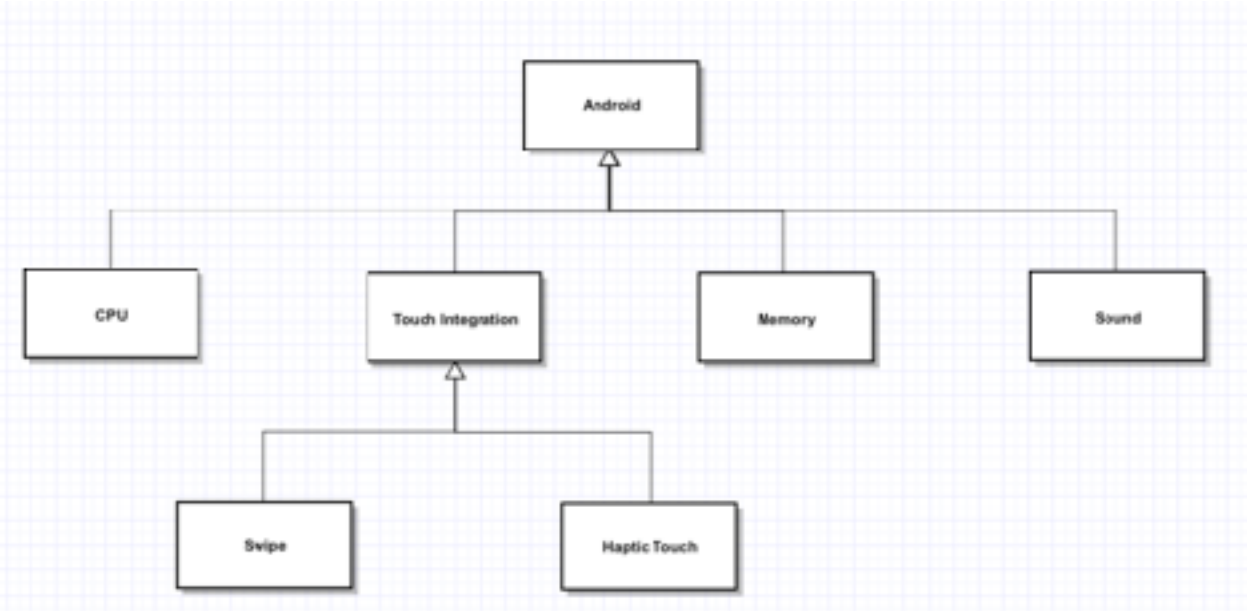
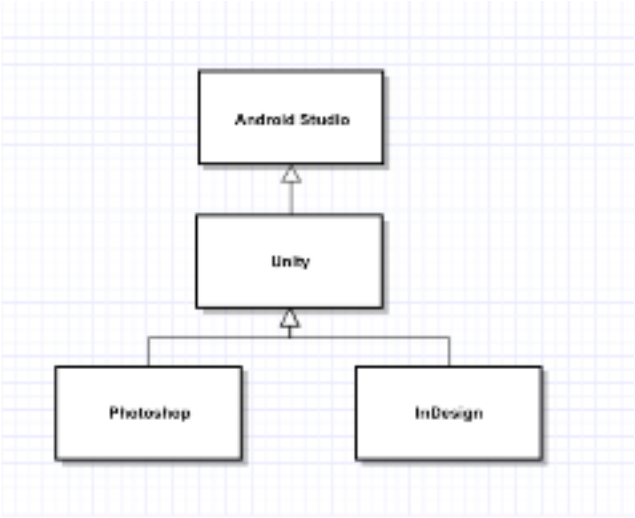


Figure 13: Software Component Class Diagram

Table 11: Hardware Description

Component



Hardware Component	Description
CPU	The CPU will process the game
Touch Integration	Touch integration will serve as the primary interaction tool with our users and the game
Swipe	Swipe will be used mainly to navigate through menus in game
Haptic Touch	Haptic Touch will be important as a way for the user to interact with the game. It will serve as the majority of the inputs associated with the games
Memory	Memory will be used to store the users scores as well as settings
Sound	The sound will play interactive sounds when the user interacts with specific actions in game as well as correspond to specific objects in game

Table 12: Software Component Description

Software Component	Description
Android	The game will then be ported into Android Studio for compatibility with Android devices
Unity	Unity will handle most of the work and most of the game will be developed under the Unity IDE
Photoshop/InDesign	Both these programs will be used to create and design our in game sprites and animations as well as any graphical models needed

4.2. Design Rationale

Our design rationale stemmed from traditional development techniques emphasizing the MVC model of development. We used android to meet the project specifications and wanted to specifically develop in Unity in order to learn the industry standard in game development as well as utilize its powerful engine to streamline the development process

5. Architectural Styles, Patterns and Frameworks

Table 15: Architectural Styles, Patterns, and Frameworks

Name	Description	Benefits, Costs, and Limitations
Unity Framework	Development will implement the unity framework	Unity normally used for 3d rendered games and has a moderately steep learning curve. Limitations would be the time it takes to learn unity as well as possibly outsourcing issues.