Don’t Die

Matt System Architecture and System Model

Hao User requirement definitions and system evolution

Carney requirement specification and Introduction

Tanaka Preface and Glossary

**Preface**

The expected readership of this document are people who know what a video game is and the basic mechanics of playing such games. The purpose of this document is to list and prioritize all requirements for a game called “Don't Die.”.

**Introduction**

We are designing a mobile application that we call Don’t Die. This is a “dodging” game. You will play a character that dodges bullets that an enemy is shooting at you. You will also be able to pick up power-ups that will help you survive.

The game will be played in landscape mode on your phone with a virtual joystick on the bottom left side of the screen to move. You will be able to change the character you are using by collecting coins that pop up randomly throughout the game. You will also be able to use these coins to unlock new backgrounds.

**Glossary**

Android Studio - android studio is the official Integrated Development Environment (IDE) for Android app development

Flappy Bird - Popular mobile game from 2012 about a flying bird dodging pipes

Sprite - 2D image used as part of graphics display

Lag - where a game freezes for a duration of time

AI - artificial intelligence

Class - in Java a class is a user-defined blueprint or prototype from which objects are created

Java - Popular class-based object-oriented programming language

Android - Android (Google) is a mobile operating system that phones and tablets can run on.

iOS - iOS (Apple) is a mobile operating system that phones and tablets run.

Joystick - mechanism for getting input from the user

**User requirements definition**

This game should be easy for people to play. Players should have at least Android 5.0 system called Lollipop for their phone to play the game. User can do these things in the list.

1. System should start up with a start menu with choices: “Play”, “Unlockables”, and “Settings”.
2. When “Play” is tapped, the system should begin to run the game.
3. When “Unlockables” is tapped, the system should take you to a screen where you can unlock characters and backgrounds with the coins picked up in the game.
4. When “Settings” is tapped, the system should take you to a screen where you can change various settings such as music and sound effect volume.
5. While playing the game, dragging the virtual joystick in the bottom right corner will cause the system to communicate with your character, making your character move.
6. When the character picks up a weapon, the new button reading “Shoot” will appear, and you will be able click on the button to fire.

**System architecture**

There will be two main packages that Don’t Die will utilize, the state package and sprite package. The state package will be used to create different states, and the sprite package will be used to create different characters in the game. The state package will consist of different classes that will be used to play the actual game, while the sprite package will consist of character interactions, items, and attributes.

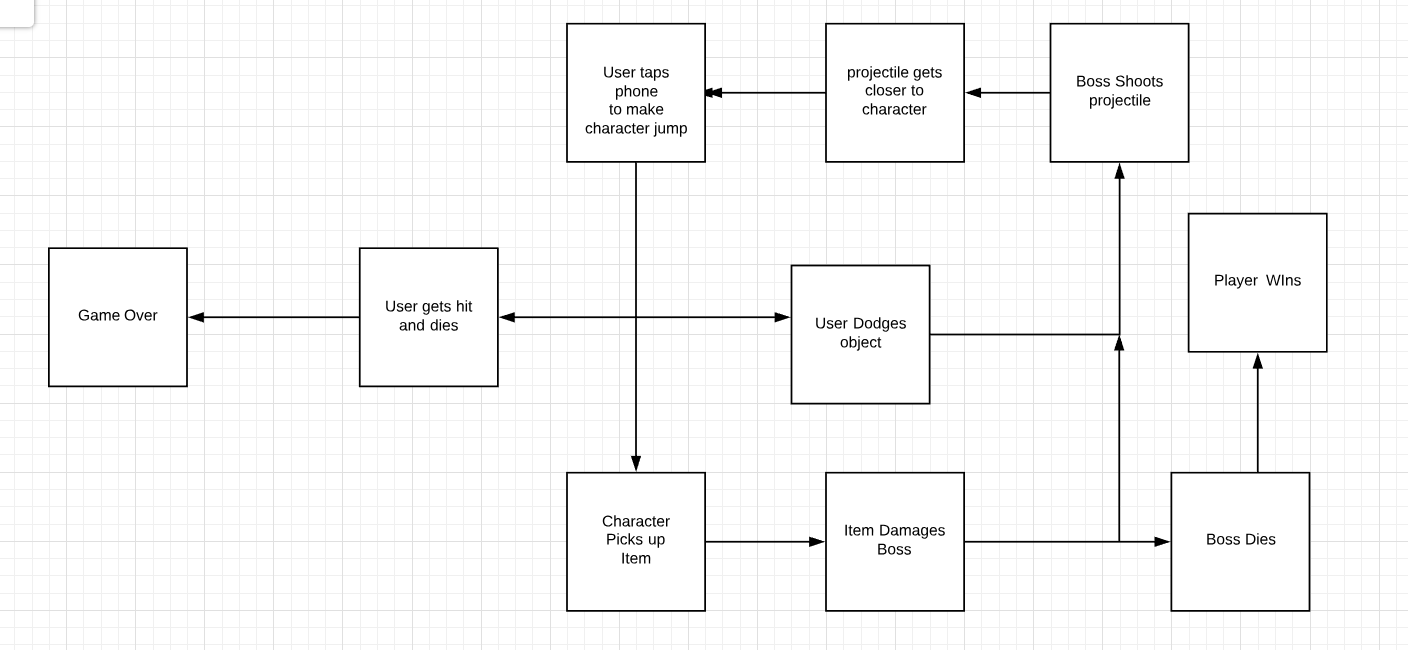
The system architecture of Don’t Die will be a stack of different states. We will have classes of different states that will be needed, such as a menu state, play state, pause state, etc. Each state will have different purposes: PlayState will be the actual game being played. MenuState will be the menu of choices. GameStateManager is essential because this class is the controller of all the states. Whatever state is on top of the stack will be the state the game is in.

The Don’t Die PlayState will track our playable character, items, coins, and an enemy character, which all fall under the sprite package. Our main Character class will have simple movement up and down and eternally progressing left to right, alongside health. The Item class will contain a list of different possible items that the playable character can use to damage the enemy. The Enemy class will contain a sprite object that will move up and down with AI that will project objects at the playable character to deal damage.

**System requirements specification**

* Must run on Android 5.0 and up.
* Should start up relatively quickly.
* Game should be developed using Android Studio.
* Game should never lag or freeze.
* System should not communicate with outside servers.
* User will not need to make an online account.
* 1 player game.
* Shall provide the ability to unlock characters.
* Must display warning that you must be 13 years or older to play.
* No in-app purchases.
* Unlimited number of coins can be collected.
* Separate classes for Boss and Characters.

**System models**



**System evolution**

* Adding more characters and weapons
* Change the style of characters
* Can accept external controller instead of using phone only
* Add multiplayer mode to the game
* Have a switch button to switch the weapon
* Make more detailed game backgrounds

For the game system, we first make the structure about the game. We can create the single player game first. Then, we can think about the game character and weapon. In the end, we may port the game to the iOS platform.

Preface: 5/5

Your preface is reasonable.

Introduction: 16/20

Your introduction is an opportunity for you to describe the system at a high level, but you could be much clearer. Are there other games similar to Don’t Die that you could compare to (with citations)?

Are there multiple levels? What makes Don’t Die unique in a world filled with casual game apps? However, this document is not for marketing purposes. There is no need to describe your process in arriving at this game choice or that it will help you develop your coding skills. It would have been useful to explain how it might be possible to complete this project in a semester.

Glossary: 5/5

Your glossary is fine, but your definition of a joystick was not very useful, especially not for a software joystick.

User requirements definition 11/15

Your user requirements give good explanations of the menus and other UI surrounding the game but not the game itself. More detail is needed.

Are lives remaining listed? Power levels? Armor levels?

A mockup showing what the game looks like would have been helpful. What will the protagonist and enemies look like? Will there be multiple levels? What kind of environments will be possible? Is it possible to win?

System architecture: 8/10

Your system architecture should have discussed larger components. What about graphics and sound? A block diagram would have been helpful.

System requirements specification: 10/15

Your system requirements specification should have been much more detailed about how the game worked. Instead, it was a list of general statements about the game, not the details of how it would work.

System models: 5/5

It’s possible that your system model should have appeared earlier in the system architecture, but it is a clear explanation of your game (although not all of the boxes and arrows quite make sense).

System evolution: 13/15

I’d like more discussion of how you might change the players and weapons, but your system evolution covered many important points. Porting the game to iOS is not reasonable, however.

Spelling, grammar, and style: 8/10

Your spelling was mostly correct, but your sentences were often awkward. Avoid run-on sentences.

Total: 81