Design Document

CSCE 361 - Summer 2017

Prepared by
<Team 3: Sasha Tenhumberg, Yue Bu, Nguyen Luu, and Ziyuan Ye>

<6/21/2017>

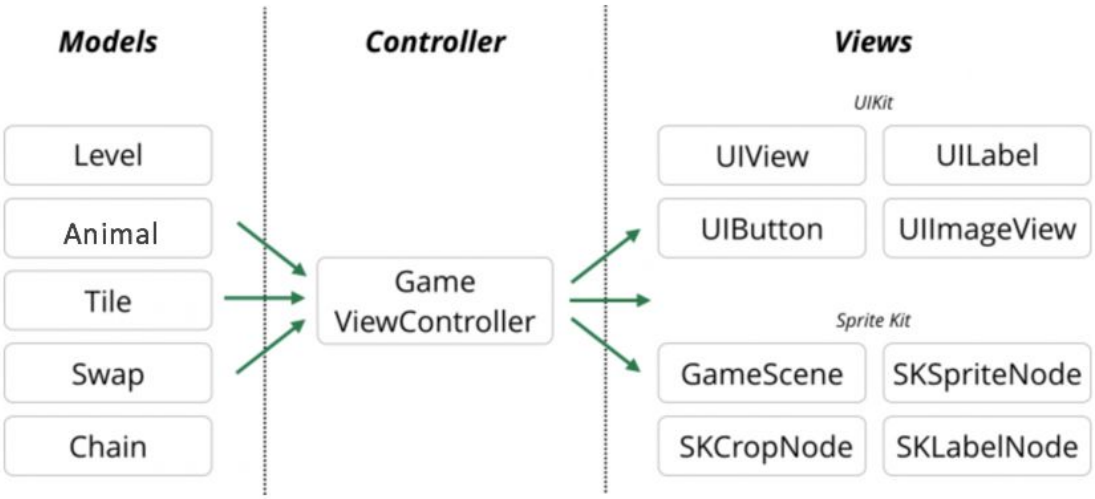
1. **Introduction**

The purpose of this design document is to display the high level architecture and entity relations for the game “Animal POP!” This document will include diagrams and explanations for the architecture as well as the relationship between the different parts in the system. The intended audience are game designers, managers, and testing engineers.

2. **Architecture**

2.1 Introduction

The high-level architectural design of the system will be an MVC model. The lowest level is the model which contain all the data for the application. The view displays the model data, and sends user actions to the controller. The controller provides model data to the view, and interprets user actions such as button clicks. The controller depends on the view and the model. For our game, we won’t be needing any database since everything is run from the game. The levels will be running from JSON files that are imported into the application.



2.2 Modules

2.2.1 Model

The sprites, level design, background, chain and swap functions are located in the model layer. The majority of the game’s data will be in this layer.

2.2.2 Controller

The Controller is the layer the majority of the logic is located. It supplies the view with the necessary output based on the model layer.

2.2.4 View

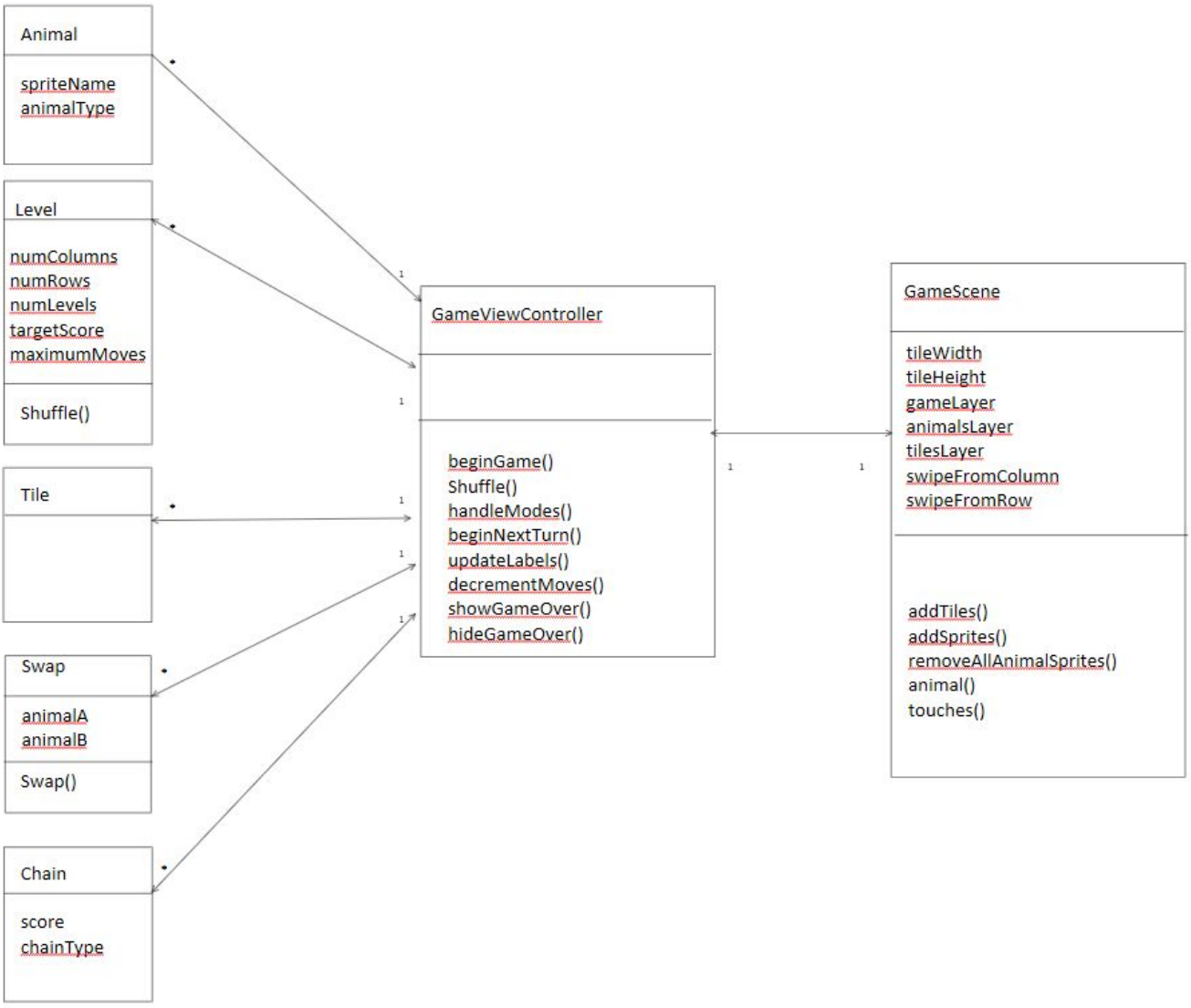
The View will be the layer the user interacts with, it has two states. The first is a menu which will allow the user to check their high scores and

move to the second state which is comprised of a grid that displays elements, a button that allows the user to shuffle the elements, and a scoreboard. In future iterations, more functionality can be added to the first state. GameScene is the main view that displays all of the game. However, it is necessary for the game to include other views such as UIView, UILabel, UIButton, UIImageView, SKSpriteNode, SKCropNode, and SKLabelNode.

3. Class Diagrams

3.1 Data Table Classes

3.1.1 Schema



3.1.2 Schema Information

The data in the database will be organized in the following tables and columns:

Animal: Holds the data for the elements of the grid. This will include the name of each sprite/animal and their types.
Level: This table holds the data for each level of the game, which includes the number of rows, number of columns, number of levels, target scores, and maximum moves.
Tile: This table should display what goes in each tile in the grid.
Swap: This table will take two sprites/animals and swap them using the swap function.
Chain: This table will determine the type of chain and the score per type of chain.
GameViewController: This controller takes the information of all the previous models and use functions to start the game, shuffle the elements (sprites/animals), handle the different matches, begin the next turn, update the elements after successful swipes, decrement the number of moves after each swipe, and display game over when the user does not successfully meet the target score after maxing out their number of moves.
GameScene: This table has the purpose of displaying the game to the user and catching the user's moves.

3.2 Class Information

Classes will be implemented in Swift and used throughout the game.

3.3 GUI Layer

The top layer of the system consists of two pages, the home screen and the game screen. When a user first opens the game, the home page will display a button so that he/she can start the game. After the user clicks on that button, it will navigates to the game page where the user will be spending most of his/her time there playing the game. The only change after that is the game grid will shuffle up/change to access a new round.