horizontal line

**Mitch Guthrie**

Game Development

University of Central Arkansas

[mitchguthrie@protonmail.com](mailto:mitchguthrie@protonmail.com)

Ink Drop - Final Report

**29th April 2020**

## Overview

Ink Drop is a new game for PC and future mobile devices. This report will give a detailed overview of the game’s design while expanding on the technical details of its development process. The format of this report is based on the original project proposal, however only including features that are now fully implemented with some commentary on mistakes made along the way.

## Table of Contents

[**Overview**](#_14mpx6a8znb7) **1**

[**Table of Contents**](#_zeaakxqbd3n4) **2**

[**Game Concept**](#_e1g3ll70ftbp) **3**

[Introduction](#_mdk7i1imycu0) 3

[Description](#_itt6re2cum5e) 3

[**Characters and Objects/Game Flow**](#_iyapjhvory67) **4**

[Sample Gameplay](#_75xnbf583jir) 4

[Gameplay Elements](#_yvrt4hm3rs8e) 5

[*Core Game Loop*](#_8ifhaf78ydic) *5*

[*Game Modes*](#_jvqjcu5j61mq) *5*

[Objects and Characters](#_wouszd6dl2vi) 6

[**Technical Details/Implementation**](#_dzd6mbpy3g43) **10**

[Level Generation and Creation](#_5o07ot9w5dgp) 10

[*Custom Level Handler - Main Game*](#_v8h0xjnocq14) *10*

[Screenshots](#_dxafgi5udjmc) 12

[Targeted Platforms](#_w22pz8l4yiwi) 14

[HUD, Scoring, and Menu System](#_ys6vylryc492) 15

[HUD and Current Level](#_6z9fa8475g9c) 15

[Menu System](#_sgmwddsydl99) 15

[**Design and Development Retrospective**](#_h2y8hibd5wu7) **16**

[Design Challenges](#_xc34swm29q1k) 16

[*Art Style/Visual Design*](#_vy12ppj4yr8o) *16*

[*Level/Menu Design*](#_yw50lv4qzhav) *16*

[Technical/Development Challenges](#_fuzvkg2uoymy) 16

[*C#/Unity Editor*](#_1tnziqi1kjhe) *16*

[**Development Roles**](#_lpy44jv5cxos) **17**

[Team Members](#_ob99xf4npkuf) 17

## Game Concept

### **Introduction**

Over the semester I was tasked with creating and reimagining a game with the Unity development platform. In the early stages of proposing this project, my original intention was to create a clone of Helix Jump with some changes on its gameplay and style. As the project progressed, my vision changed, along with my development process. Looking back, I focused my energy on optimization of workflows and programming processes more than on the creativity of the game’s design. I am still pleased with the result and learned many valuable game development skills along the way. While the semester may have concluded, I will continue my development of ‘Ink Drop’.

### Description

Ink Drop is a deceptively simple arcade style game currently released on Windows PCs. Taking inspiration from Helix Jump, your task is to help a drop of ink maneuver it’s way down a tower by rotating the game world and avoiding any obstacles along the way. The ink drop will safely bounce on standard platforms, but there are sponge platforms on the tower that won’t hesitate to absorb the little drop. Each platform will have an opening, by rotating the tower you can help the ink find a pathway down until it reaches the bottom. Completing each level will gradually increase the difficulty in forms of removing platforms and increase the number of obstacles in your way. At the bottom of each level there will be a ‘safe platform’ that marks the level as complete and moves the player onto the next stage. An arcade style point system is displayed on the top of the game screen; clearing platforms increases your score while hitting an obstacle resets it to 0. The game will store the highest score in the registry so it won’t erase when exited.

## Characters and Objects/Game Flow

### Sample Gameplay

After selecting ‘PLAY’ from the main menu, the player will show the initial setup for the first game level. Each game starts with the drop of ink located at the top of a tower of unknown length (to the player). The player can use the mouse, by clicking and dragging, to rotate the tower to guide the ink drop towards the gaps in the platforms and help it descend the tower. Some of the following platforms will have sections of sponge that the ink cannot touch or the level would reset. Each level increases difficulty by removing safe platforms, increasing sponge blocks, and adding obstacles. By dropping down a platform, the player’s score will increase, along with the ink drop’s speed. If the ink comes in contact with a sponge block or obstacle, the current level is reset and the score drops to 0. The information: current score, current level, and best score are displayed on the screen at all times. Once the ink makes it to the bottom of the level and is successfully guided into the safe platform, the player will receive all the points gained that level and will be directed into the next level. The camera perspective will be static and will be centered on the ink at all times.

### Gameplay Elements

#### *Core Game Loop*

This game is extremely fast paced and should keep the player hooked on trying to always do better than previous attempts. The player will immediately start the level over at the top of the tower each time the ink hits a sponge block and the points will be reset. Once successfully completed, they will unlock the next level, with added difficulty. The game loop will only end when the player chooses to exit the app or leaves to visit the main menu.

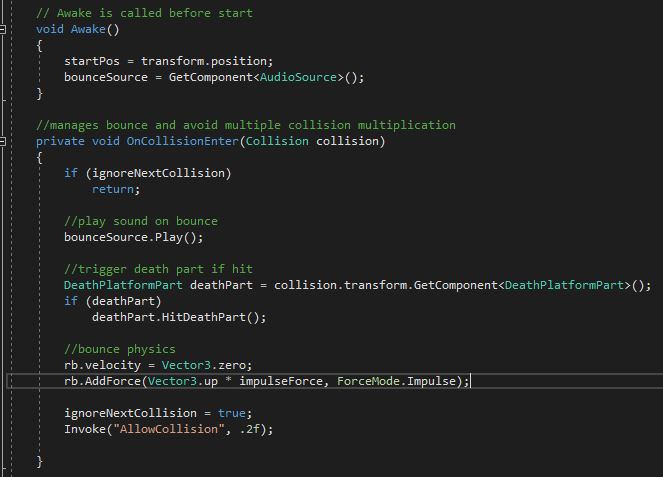
#### Game Modes

There is only one main mode/loop, however after winning a level, the player will have the option in the main menu to start from that level instead of all the way from the beginning each time. Currently there are 30 standard levels that increase in difficulty. The levels are set to change as development progresses. I have created a custom level file system that allows quick editing and deployment of levels without modifying the main game scene. The current difficulty is sharp so it will take the player a sizable amount of time to complete the game.

### 

### Objects and Characters

* **Ink Drop**: Fully implemented, has bounce physics and only moves on the Y-axis, affected by gravity.



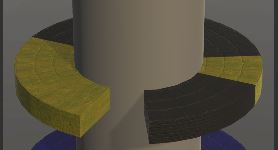
* **Tower**: Fully implemented, can be rotated using the mouse, contained within the Stage entity that handles platform spawning and calculations.

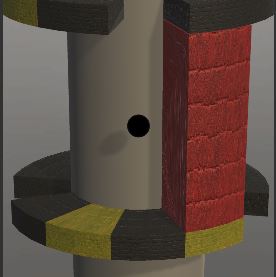


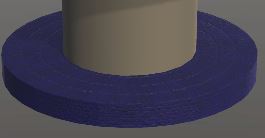
* **Normal Platform**: Fully implemented, created with 12 sections that are spawned and controlled via the stage controller. Two static normal platforms are the Top and Goal platforms.

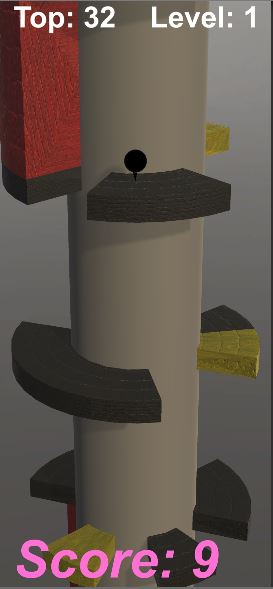


* **Sponge Platform**: Fully implemented, normal platform with added script that changes texture and resets ink drop on collision.



* **Obstacles**: I created a system branching from the platform editor, to allow the definition of new ‘Wall’ traps that will follow the same generation logic and editability as the hostile sponge blocks.
* **Safe Platform**: Due to lack of detailed custom assets, the concept of the ink bottle has been replaced with a ‘completion platform’ at the bottom of each level.



* **Background Scene**: Fully implemented using skybox currently. Depending on which styles the player has unlocked, the background will feature various scenes and settings, these do not impact gameplay but are a cosmetic choice.
* **Menu Elements**: The HUD contains an indication of which level the player is currently on along with current score and high score. The main menu has two buttons for starting the game and options. Options sub menu implemented.

## Technical Details/Implementation

### Level Generation and Creation

#### Custom Level Handler - Main Game

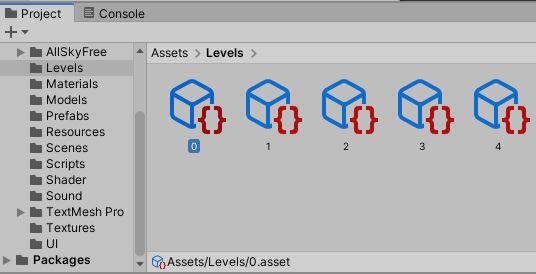
The custom level handler and stage controller work together to allow setting attributes such as: number of levels, number of platforms, number of elements on each level. This means creating new levels only takes a few seconds, instead of having to hand-place every asset.

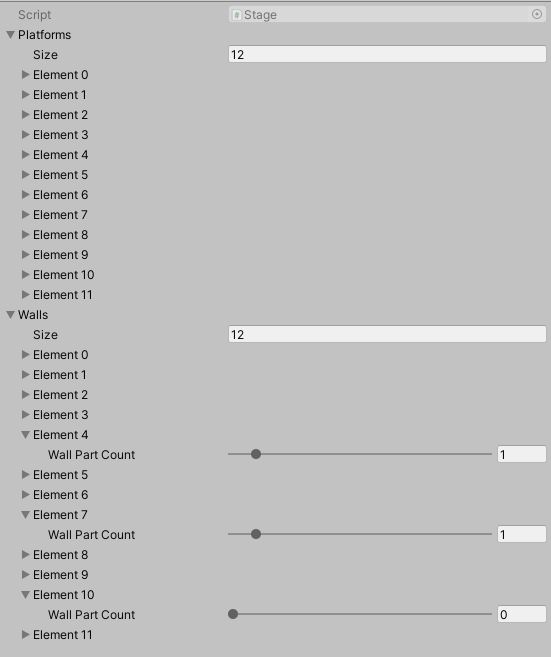
*Stage:* This script allows for the creation of custom ‘level’ files that contain attributes that can be processed by the StageController script. The stage script sets the max amount of sections in a platform to be 12, with sponge platforms not allowed to go past the max currently spawned regular platforms. The platforms are held in a List data structure that adds up to a full platform.

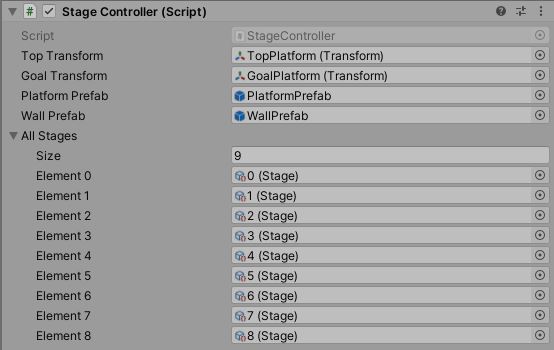
*StageController:* This is a more complex script that is added onto the Stage entity group containing: Top and Goal Platforms, the tower, and spawnable platforms. On Awake, the script sets the current rotation of the tower and calculates the distance between the top and goal platforms. In the update function, the tower position is used to implement user rotation with the mouse. The custom LoadStage function reads from the array of custom levels and the attributes of each level. These attributes determine how many platforms are on each level, how many sections of each platform are active, and how many active sections are sponge sections. The position of the spawned platform sections is random while the number of platforms is static. This script is a modified version of a system I found from the TutorialsEU youtube channel so I will credit them in the references section.

*Obstacle System:* In a similar fashion as the platform generation, this class handles the creation and customization of obstacles. Obstacles are based on prefabs, the newest being a ‘wall’ that spawns perfectly between two platform levels and will cause the player to restart if they collide. These walls follow the same generation algorithm as the platforms but with added semi-randomness as to not appear in the exact same position as the missing platforms. Since this script is tied to the custom level files, it’s easy to set: how many obstacles will be in a given level and where they will spawn on the z-axis. This is a flexible system, which means if I want to add more types of obstacles, all I would need to do is create a new prefab and duplicate the script.

### Screenshots

[Figure 1a. The custom level files. Created by right-clicking and adding a ‘level’ in unity]

[Figure 1b. The setting of level attributes within the stage file]

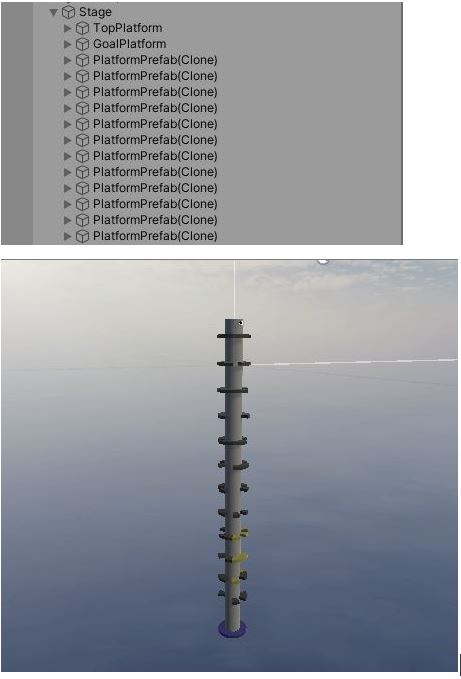


[Figure 1c. Loading level files into the ‘stage’]





[Figure 2a. The default level before the game is started]



[Figure 2a. The level, fully populated, after the game is started]

### 

### Targeted Platforms

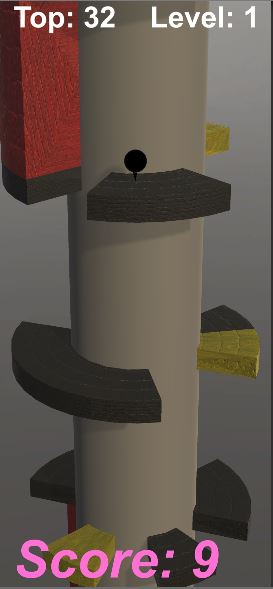
Upon official release the game will be fully compatible and playable on PC. Since I do not have an apple developer license iOS compatibility will come along in the future.

### HUD, Scoring, and Menu System

#### HUD and Current Level

The HUD contains the players current score at the bottom of the screen and their best score and current level at the top of the screen. This was created with Unity’s canvas and text entities that are controlled with a custom UIManager script. This reads from the set of custom level files and displays the player’s current level accordingly. This makes it much more clear that when the player hits an obstacle, only the current level restarts rather than the entire game.

#### Menu System

The menu system uses Unity’s canvas system to create a simple menu background, title text, and buttons. The MainMenu entity contains the nested PlayButton entity and OptionsButton, allowing for easy creation of more buttons. All menu elements are handled by the MainMenu script that currently switches to the ‘Game’ scene when the play button is clicked.

## Design and Development Retrospective

### Design Challenges

#### Art Style/Visual Design

The biggest hurdle to overcome with the visual design aspect of this project was learning the unity graphics pipeline. This is my first time using Unity and I had to learn how to: create assets, handle shaders and textures, optimize rendering for distant objects and manage Level of Detail to run smoothly on all systems. This caused some roadblocks and unfulfilled features.

#### Level/Menu Design

This proved not to be as tough as I imagined in the project proposal, after establishing an in-house level creator it became much easier to create levels. However, since my assets were limited, the levels still feel too homogeneous.

### Technical/Development Challenges

#### C#/Unity Editor

I am well versed with Java and similar object oriented programming languages, but C# was and is still new to me and I had to really brush up on my OOP ability to achieve some of the ideas I had. Unity itself was fairly intuitive so the only issues were with learning the workflow. Some planned features, animation and physics destruction, were sadly out of my grasp during the allotted amount of time I had this semester.

## Development Roles

### Team Members

* Mitch Guthrie - 001215440 - Lead Developer