

**Maintenance Document**

|  |
| --- |
|  |

Sample document: http://www.techstreet.com/direct/SWM\_samples.pdf

.

# Introduction/Something

The software is a 2D game made with Unity 2017.3.0f3. The game consists of a number of levels in addition to the menu screens. The game is modelled after the 1982 game Jungle Hunt. This chapter gives an overview of the software structure.

The game starts with menu screens. During these screens the difficulty and the players name are saved into the DataContainer\_Character script. This script holds all the information related to the player’s success in the game: current level, scores and the number of lives.

Each level is generated dynamically using the corresponding LevelGenerator script. This script uses prefabs and graphics with the information saved into DataContainer\_Character to create the level. Renaming or changing graphics or prefabs should be changed accordingly in the generator scripts.

The Player-script is used to move the player and change the players state and animation. PlayerCollision takes care of the different collision and triggers that happen when the player hits non-lethal objects.

The game uses Unity’s Input Manager. This ensures that the game is playable with different control schemes and developers can change the bound buttons easily. The menu’s can be navigated with mouse, keyboard or another device. Player can be controlled using keyboard or similar device.

The software has built-in unit tests for SceneManager and DataContainer\_Character scripts. These tests can be run be using the Unity’s own Test Runner. In addition to these tests, the software has been tested manually. In normal use the game should run as intended.

-Known bugs or problems

# Placeholder for more indepth guide

## Scene\_Manager

Scene\_Manager (note that this is not the same as Unity’s built-in SceneManager) is a large part of preserving continuity in the game. It handles Scene Transitions and the general flow of things within the game.

Scene\_Manager is hosted in an “Overlay Canvas” object which allows it to draw UI objects on screen without need for camera references or other constraints. This allows each Scene to be designed uniquely while still maintaining similar features, such as UI.

Scene\_Manager is responsible for maintaining the following main game features:

* Sound (via SoundSystem component)
* User Interface and Transitions.
* Input Events (using Unity’s EventSystem)
* Loading levels and providing ways to load levels.
* Generating and Maintaining Level Order.

The Scene\_Manager is generated on boot-up when loading to the main menu. An additional check is performed to ensure no additional copies are created. The manager only has two main functions to call: ChangeScene(int scene) and NextLevel(bool scoreboard) the former allows any scene to traverse to another without regard to player condition etc. the latter allows levels to call the next level without caring what level is coming next. The following is an outline of main functions that are travelled through when changing scenes:

**Next Level** – *Generates a level order if not existing and calls the next one.*  
**Change Scene** – *Starts change to next scene, starts by initiating a transition event to which the following is always subscribed to.*  
[Scene Change Component - Image] **Drop Image** – *Drops a transition screen.*  
[Scene Change Component - Image] **Score Board Completion** -> **Button Call** – *If scoreboard Boolean is set to true, a scoreboard is shown and a button to continue given, otherwise skipped.*  
[Scene Change Component - Image] **Wait For Completion** – *Begins an asynchronous load and waits for it to finish, then raises the transition screen.*  
**Start Load** – *Called when Load starts, used for fade outs, hiding UI etc.*  
**Finished Load** – *Called when Load finishes, used to play new BGMs and show UI elements etc.*

Start Load and Finished Load functions have various conditional checks that allow it to hide, show UI components and play the correct BGM for each level. The entire system uses Unity’s BUILD INDEXES to determine what screen is being presented, including the Change Scene function.

Player (Player Collisions?)

-States and triggers