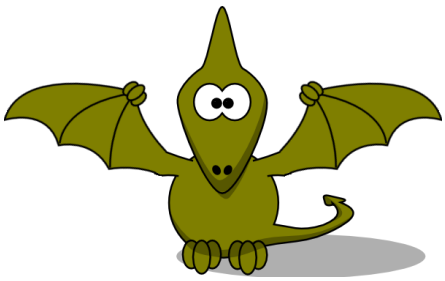
Monster Runner Game

Standard Version for Mobiles

plasticinegames.com



Introduction

Thanks for purchasing the endless monster runner template game from us. We hope you enjoy the template and unleash your creativity and produce new exciting runner type games. Monster Run has been scripted using C# and porting to JavaScript/UnityScript should not be difficult. Please remember the game is a template and if you wish to deploy to it to any popular app stores, it may be prudent to change the graphic and audio assets and introduce new gameplay to differentiate from any other developers offerings.

Please note, we have not submitted the template to any app stores. Play the template online here:

https://dl.dropboxusercontent.com/u/32619325/MonsterRunTemplate/WebPlayer/monster_run_v10/monster_run_v10.html

The Game

The game is based on the popular endless runner type games and consists of a monster character running down various backgrounds and avoiding baddies and obstacles in the players monsters path. The goal is to run as far as you can while collecting cookies or eggs to score points while avoiding the bad monsters. The game is designed to run on iOS and Android mobile devices and should work on other devices too with little minimum UI changes.

The game demonstrates most aspects of a commercial game including sound effects, background music, displaying a splash screen, high scores, pause overlay and customisation of themes.

The Structure of the Monster Run

The structure is composed of gameobjects, prefabs, materials, textures and audio files grouped in parent-child hierarchies. The player monster character is a child node of a main camera gameobject with an orthographic projection ideal for 2D games without any depth perspective. The game uses primitive colliders e.g., box colliders with the trigger property enabled. The main monster is the only object to contain a rigid body as collisions are detected between the monster and other gameobjects such as enemies, powerups and food.

The files will be found in the project asset folder window. A good approach to writing code in scripts is ideally not to duplicate code (DRY don't repeat yourself) so could be easily be reused/modified and results in easier to maintain code too. The project uses iTween by PixelPlacement for all tweening operations. We believe once you understand iTween you may want to create the effects by writing your own code too. iTween is powerful and demonstrates its power with a single line of code.

The enemy, player, food and environment gameobjects are enabled and disabled depending on which character and theme is selected in the theme settings screen (gameobject).

The game has lots of comments in the code to help understand the logic and flow of the scripts.

There are many different strategies to create an app that runs on multiple mobile devices with different aspect ratios. Monster run target's a screen aspect ratio is 4:3 and scales GUI textures. Other mobile devices may have other aspect ratios such as 16:9 (See appendix A).

The scripts have been designed to be loosely coupled with minimum script to script communication or message passing.

Basic Structure of Monster Run

1. Splash Screen loads and displays UI elements for 3 seconds (Fades)

2. Splash Screen then loads the Game Scene (main menu)

3. Game Scene loads and displays the game menu

Scenes

1. SplashScreen - Display your logo/branding in this screen
2. GameScene - The game scene (including sub screens)
3. Highscores - Display the top scores and distance ran

Overview of Game Objects in GameScene

Background

The Background gameobject is a parent for all its child nodes or game objects, responsible for displaying the scenery gameobjects and scrolling them upwards (+ Y axis) e.g., checkpoint, trees, grass clumps etc.

Enemy

The Enemy gameobject is a parent for all its child nodes or game objects, responsible for displaying the enemy gameobjects, detecting collisions and scrolling them upwards (+ Y axis) e.g., GingerbreadMan, Enemy1. It also contains the overlay displayed for each theme selected by the player in the theme settings screen.

Food

Identical to the previous gameobjects the Food gameobject is also a parent for all its child

nodes or game objects, responsible for displaying the food gameobjects, detecting collisions and scrolling them upwards (+ Y axis) e.g., Biscuit1.

Main Camera

The MainCamera gameobject is a parent for all its child player monster nodes and is responsible for displaying the selected player monster gameobject, detecting collisions with the Enemy gameobjects.

ThemeSettingsGUI

The ThemeSettingsGUI container gameobject contains a single child gameobject which has a child node and is responsible for displaying the themesettings screen and other GUITextures for selecting a player character, selecting a theme, accelerometer sensitivity, SFX and background music.

TOPGUI

The TOPGUI gameobject is responsible for displaying the player score, distance ran and health. The child nodes are also responsible for displaying the main menu i.e., Play, ThemeSettings and HighScores buttons. Textures are displayed using the GUITexture gameobject.

Overview of Game Objects in Highscores

Background

The Background parent gameobject is a duplicate copy from the gameobject in the GameScene. it's role is to dynamically change the overlay depending on the theme the player has selected from the ThemeSettings.

BlankObject

This gameobject is a container to hold a GUITexture and script to update the scores and distance and process touch/keypress events and return to the GameScene.

Top5Scores

The gameobject is a parent for its gameobjects (distance, score and position GUITexture). The score and distance GUITextures are displayed using their default values and are overwritten by the script in the BlankObject.

Common & Important Scripts Explained

An overview of pertinent scripts that we feel are important to gain an understanding of the structure of the game.

ScrollBackground (common): Declares three public types (Transform + Float) and scrolls each object at the set Float value along the +Y axis.

AnimateObject (common): Declares three public types (Float) and scrolls the attached object at a set random interval along the +Y axis.

Checkpoint: Declares two public types (Float) and scrolls the attached object every 1000 units along the +Y axis.

GenericController (common): Declares two public types (Gameobjects) and depending on the Global.playerThemeType value either activates or deactivates the parent containers for EnemyType1.

BadBiscuit (common): Declares seven public types (Texture x2, Floats x3) and performs scrolling, collision testing, bounds checking, animation and displaying the gameobject.

MonsterAnimateController (common): Declares eleven public types (Texture2D x2, Floats x3, Gameobject x3, AudioClip x3). The script is responsible for getting player input (touch/keyboard), collision testing, bounds checking, animation, pausing the game, playing sound effects and displaying the gameobject.

New Ways to Extend Monster Run

- Additional foreground textures
- 3D perspective
- Integration with OpenFeint & Facebook
- Add more themes, characters and background scenery
- Add different characteristics to each player and/or enemy game objects (script properties)
- Extra Life and Invincibility for 5 seconds
- In-app purchases
- Add more enemy gameobjects

The next version of the template may include some of the features described above. Please let us know what features you would like to see. Thank you for your support in purchasing our asset and good luck with your games :)

F&Q

How does the background music continue playing when changing scenes? A BackgroundMusic gameobject has an attached script (Singleton) that calls a method DontDestroyOnLoad. This object persists across scenes and continues to play the background track.

Why do you use iTween? It's a popular tweening tool among developers and a common goal is ideally not to reinvent the wheel in other word to use plugins to increase your development speed to hasten your game development.

I am totally stuck please help? Please send us an email explaining the problem and describe your environment i.e. Unity Version, Operating System, Target Device OS, Target Device Model, Target Device Screen Resolution and any other information that may be useful.

Can I sell the app on Android and/or iOS app stores using the current assets? Yes you can though we would recommend tailoring the game to your specific needs by adding new graphics and sound assets.

How can I make make money with the app? You could sell the app at a premium price or for free and using advertising and/or in-app purchases.

Why did you write the monster template? As games makers we needed a route to fund our 'main' game so we took time out to create this template to demonstrate how to create a top down runner game.

Support

assetstoresupport@plasticinegames.com – All emails normally answered within 24 hours.

plasticinegames.com - Unity Asset Store and game repository for BB/Android and iOS games

NB: We can only support the existing template and not any game you are creating from it.

Credits

Art

http://kevz10.deviantart.com/art/Cute-Monsters-and-Nice-Sprites-310388727

www.clkr.com

www.openclipart.com

http://www.freeclipartnow.com

Online Drawing Tool

http://www.draw.io/

Sounds & Music

http://www.freesound.org

Packages Used

Pixelplacement.com (Tweening)

http://itween.pixelplacement.com

http://www.youtube.com/watch?v=qE5hpp4YaH4 (def watch this if you are new to iTween)

http://www.youtube.com/watch?v=z_yynd8iLeM

Unity3D Tutorials and Links we like

Multiplayer FPS Tutorial

http://snaphackers.blogspot.co.uk/2013/02/beginners-to-pro-tutorials-of-unity-3d-4.html

Gems for Unity

http://www.unitygems.com

All about Unity3D Textures

http://docs.unity3d.com/Documentation/Manual/Textures.html

Appendix A - Building for Different Screen Aspect Ratios

The monster run template targets a screen aspect ratio of 4:3 and uses a strategy of placing a larger foreground texture in a gameobject. With this technique the game can be played on devices supporting most common aspect rations for mobiles and tablet with little or no code changes to the game. The disadvantages include the game does not fill the entire screen however if designed well, it should have minimum impact on the game play.

The template has been tested on physical devices with aspect ratios of 4:3, 3:2: 16:10 and 16:9 on both iOS and Android platforms.

The GUITextures in the game should be dynamically positioned and scaled for each aspect ratio to ensure they display correctly.

Common Aspect Ratios

Aspect ratio	Resolutions	Example Devices
4:3	320x240	Android devices
	1024x768	iPad 1, iPad 2
	2048x1536	iPad 3
3:2	480x320	iPhone 3GS and lower, Android devices
	960x640	iPhone 4, iPhone 4S
16:10	800x480	Android devices, WindowsPhone7
	1280x800	Android tablets like Google Nexus 7, Samsung Galaxy Tab 10.1, Motorola Xoom, Asus Eee Pad Transformer
17:10	1024x600	Android tablets like Samsung Galaxy Tab 7
16:9	640x360	Symbian3 devices like Nokia C7
	854x480	Android devices, MeeGo N9

	1136x640	iPhone 5

NB: Monster run has been tested on the following devices: iPhone 4 (AR 3:2), iPad 1 (AR 4:3), HTC Mobile (AR 16:9).

Appendix B - Adding a new player character

Please follow the steps described below:

1. Create the graphical textures (3 frames) for your new player character at 128x128. One frame should be static image and the remaining 2 frames should be animated e.g., legs moving up and down.

2. Find the 'Main Camera' parent gameobject and Copy and paste an existing PlayerMonster and rename it PlayerMonster4 and set the textures to your new textures.

3. Add another public variable for a gameobject type in the **MonsterController** script. See the script for more details.

4. Drag the newly created PlayerMonster4 gameobject onto the public gameobject variable in the Main Camera gameobject

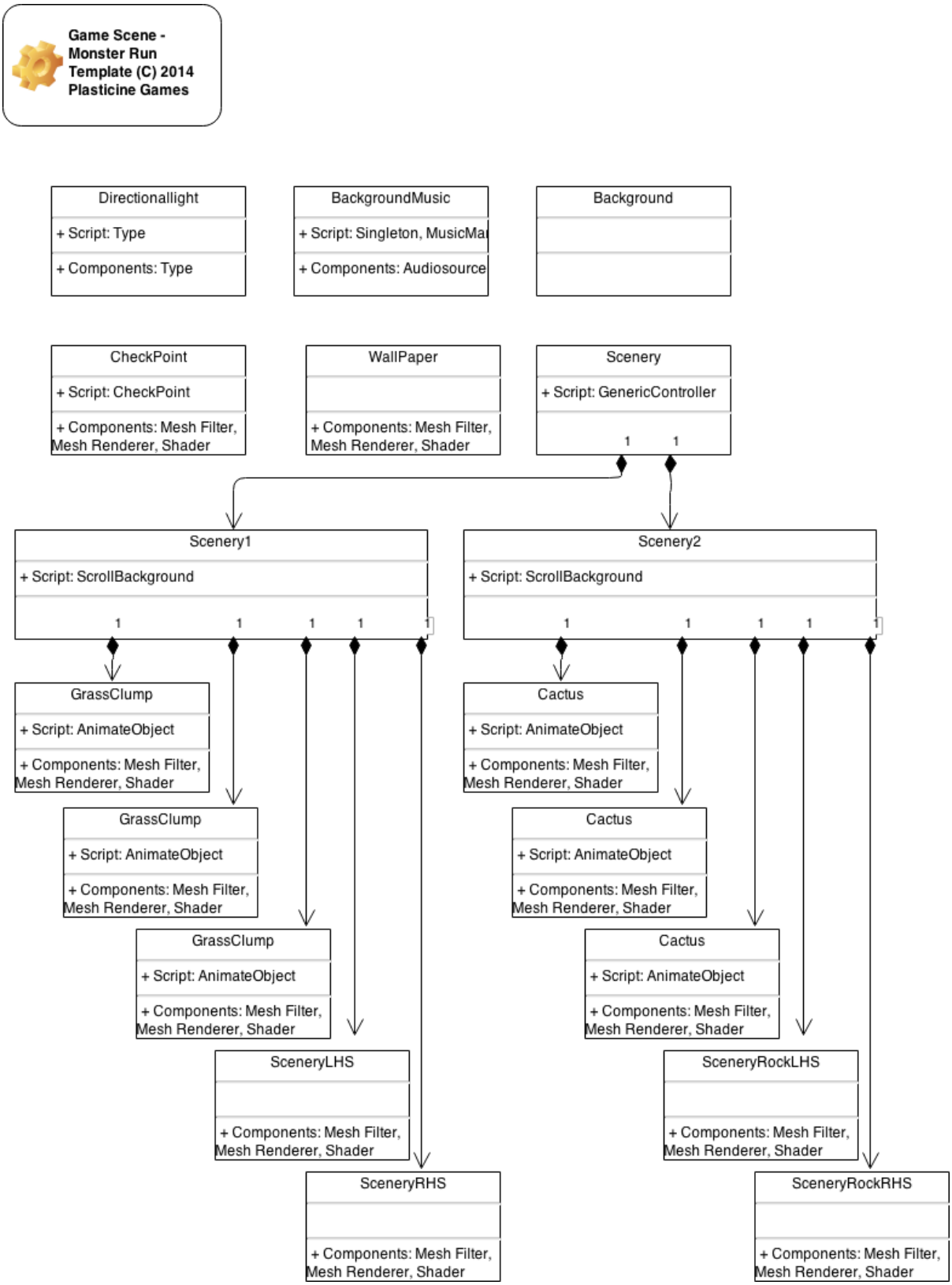
5. Drag the new static texture for your player character onto Texture 4 in the **ThemeOptionSelection** script and change the public variable (Number of Images) to 4.

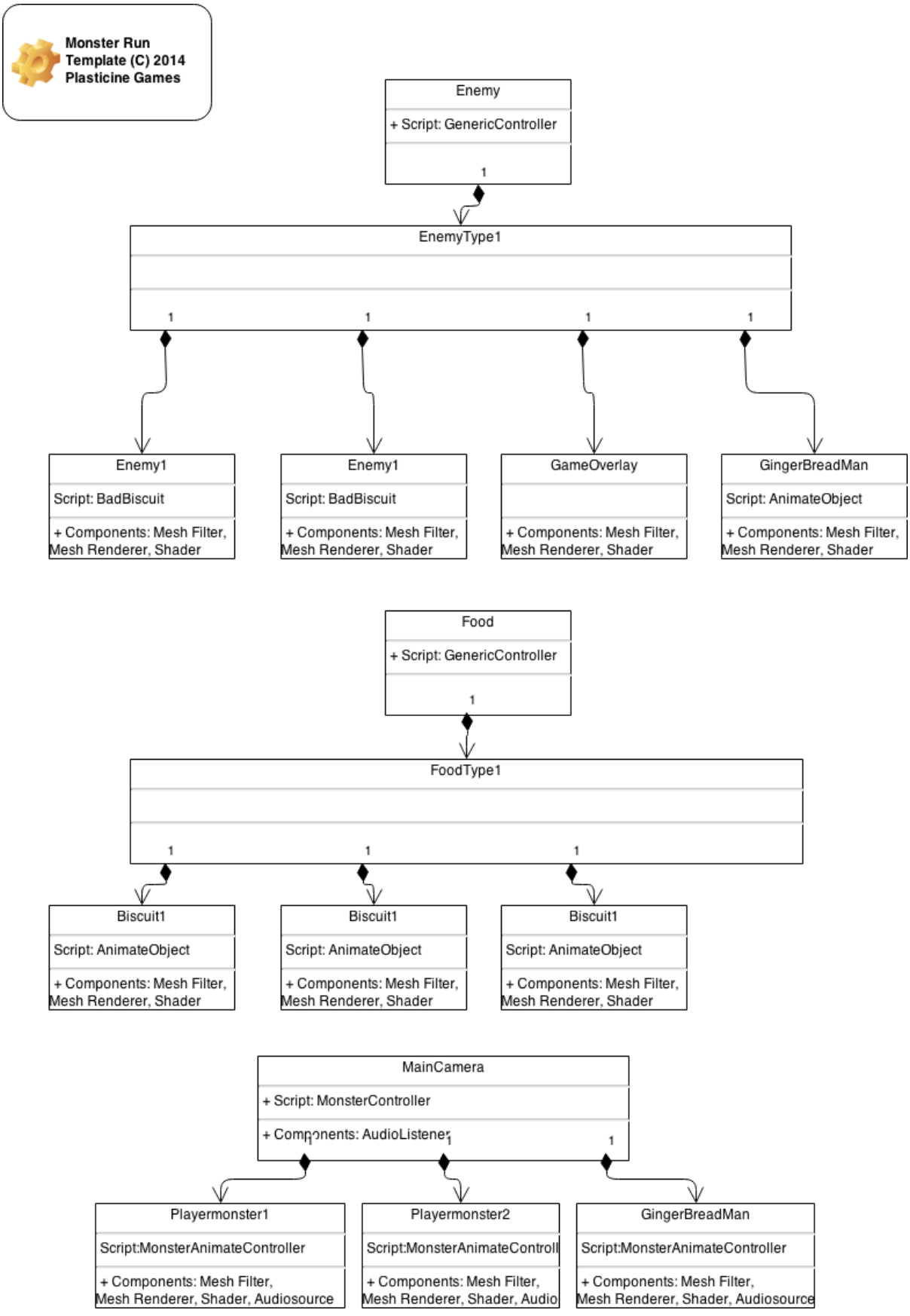
Your new character should be selectable from the theme settings and should be displayed when the game is played.

Appendix C - Hierarchy of Gameobjects in GameScene

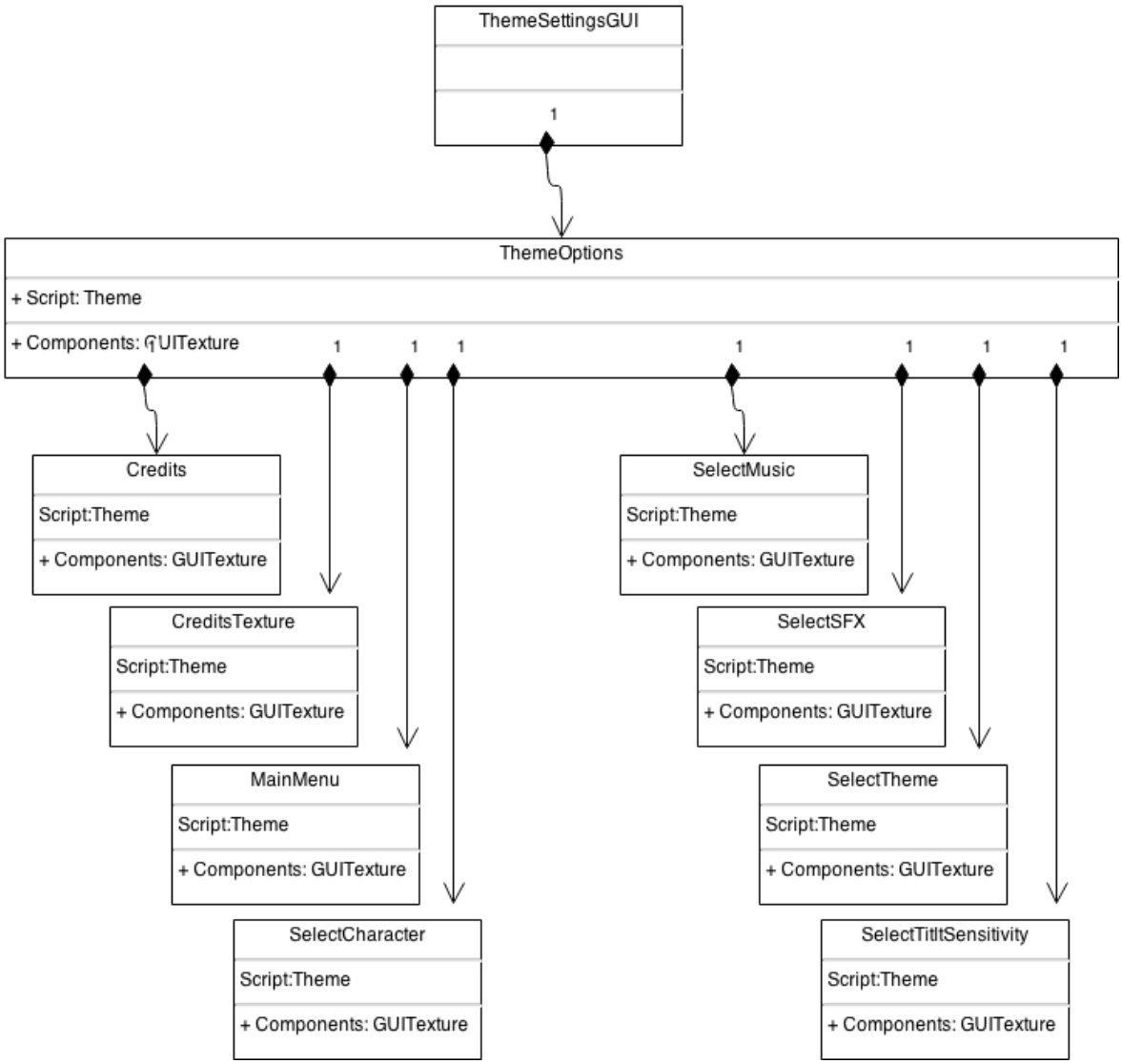
The following diagram shows the relationships between parent and child nodes with descriptions of the main components in each object. The hierarchy view should make it easier to understand how the template is built and how the gameobjects function.

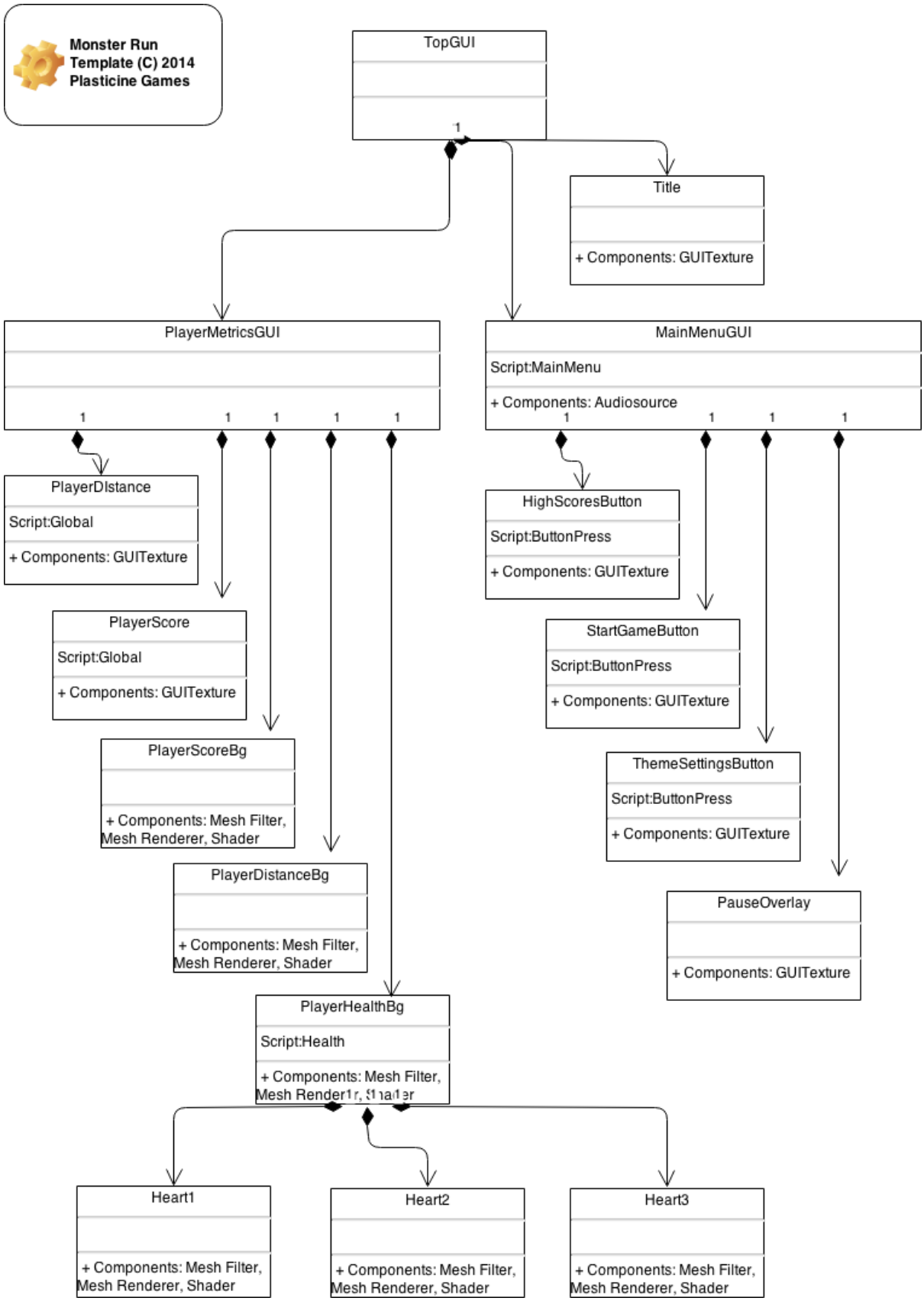
To introduce new scenery to the background ideally you should duplicate the Scenery2 child gameobject and replace the materials in the child gameobjects with your new material (including a texture). You should ideally rename the new scenery gameobject Scenery + scenery number e.g., Scenery3 and rename the child objects with descriptive names similar to Scenery2.


Game Scene - Monster Run Template (C) 2014 Plasticine Games
Directionallight
+ Script: Type
+ Components: Type
BackgroundMusic
+ Script: Singleton, MusicMa
+ Components: Audiosource
Background
CheckPoint
+ Script: CheckPoint
+ Components: Mesh Filter, Mesh Renderer, Shader
WallPaper
+ Components: Mesh Filter, Mesh Renderer, Shader
Scenery
+ Script: GenericController
1
1
Scenery1
+ Script: ScrollBackground
1
1
1
1
1
Scenery2
+ Script: ScrollBackground
1
1
1
1
1
GrassClump
+ Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
GrassClump
+ Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
GrassClump
+ Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
SceneryLHS
+ Components: Mesh Filter, Mesh Renderer, Shader
SceneryRHS
+ Components: Mesh Filter, Mesh Renderer, Shader
Cactus
+ Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
Cactus
+ Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
Cactus
+ Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
SceneryRockLHS
+ Components: Mesh Filter, Mesh Renderer, Shader
SceneryRockRHS
+ Components: Mesh Filter, Mesh Renderer, Shader


Monster Run
Template (C) 2014
Plasticine Games
Enemy
+ Script: GenericController
1
EnemyType1
1
1
1
1
Enemy1
Script: BadBiscuit
+ Components: Mesh Filter, Mesh Renderer, Shader
Enemy1
Script: BadBiscuit
+ Components: Mesh Filter, Mesh Renderer, Shader
GameOverlay
+ Components: Mesh Filter, Mesh Renderer, Shader
GingerBreadMan
Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
Food
+ Script: GenericController
1
FoodType1
1
1
1
Biscuit1
Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
Biscuit1
Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
Biscuit1
Script: AnimateObject
+ Components: Mesh Filter, Mesh Renderer, Shader
MainCamera
+ Script: MonsterController
+ Components: AudioListener
1
1
1
Playermonster1
Script:MonsterAnimateController
+ Components: Mesh Filter, Mesh Renderer, Shader, Audiosource
Playermonster2
Script:MonsterAnimateControll
+ Components: Mesh Filter, Mesh Renderer, Shader, Audio
GingerBreadMan
Script:MonsterAnimateController
+ Components: Mesh Filter, Mesh Renderer, Shader, Audiosource

Monster Run
Template (C) 2014
Plasticine Games




Monster Run
Template (C) 2014
Plasticine Games
TopGUI
1
Title
+ Components: GUITexture
PlayerMetricsGUI
1
1
1
1
1
PlayerDIstance
Script:Global
+ Components: GUITexture
PlayerScore
Script:Global
+ Components: GUITexture
PlayerScoreBg
+ Components: Mesh Filter, Mesh Renderer, Shader
PlayerDistanceBg
+ Components: Mesh Filter, Mesh Renderer, Shader
PlayerHealthBg
Script:Health
+ Components: Mesh Filter,
Heart1
+ Components: Mesh Filter, Mesh Renderer, Shader
Heart2
+ Components: Mesh Filter, Mesh Renderer, Shader
Heart3
+ Components: Mesh Filter, Mesh Renderer, Shader
MainMenuGUI
Script:MainMenu
+ Components: Audiosource
1
1
1
1
HighScoresButton
Script:ButtonPress
+ Components: GUITexture
StartGameButton
Script:ButtonPress
+ Components: GUITexture
ThemeSettingsButton
Script:ButtonPress
+ Components: GUITexture
PauseOverlay
+ Components: GUITexture