**List of scripts**

*Player*

Playercontroller.cs

Mouselook.cs

WeaponController.cs

*Entities*

EnemySpawner.cs

EnemyBehaviour.cs

PiggyBehaviour.cs

*Environment*

PickUpScript.cs

InteractScript.cs

ConversationManager.cs

ShopManager.cs

UpgradeManager.cs

FastTravelSaveManager.cs

*Statistics (Don’t destroy on load)*

StatsManager.cs

QuestsManager.cs

PlayerSpawnLocator.cs

InventoryManager.cs

*User Interface*

UIManager.cs

PauseMenuManager.cs

MainMenuManager.cs

LoadController.cs

CreditsController.cs

*Functionality*

PlaySoundAtPoint.cs

DontDestroyAtLoad.cs

**Classes**

Questclass.cs

ItemClass.cs

EnemyClass.cs

**Language Structure**

Language: C#

Comment lines before functions.

Brackets on new lines

No hard-coding

Private Component variables start with underscore

Variables start with lower-case letter

Functions start with upper-case letter

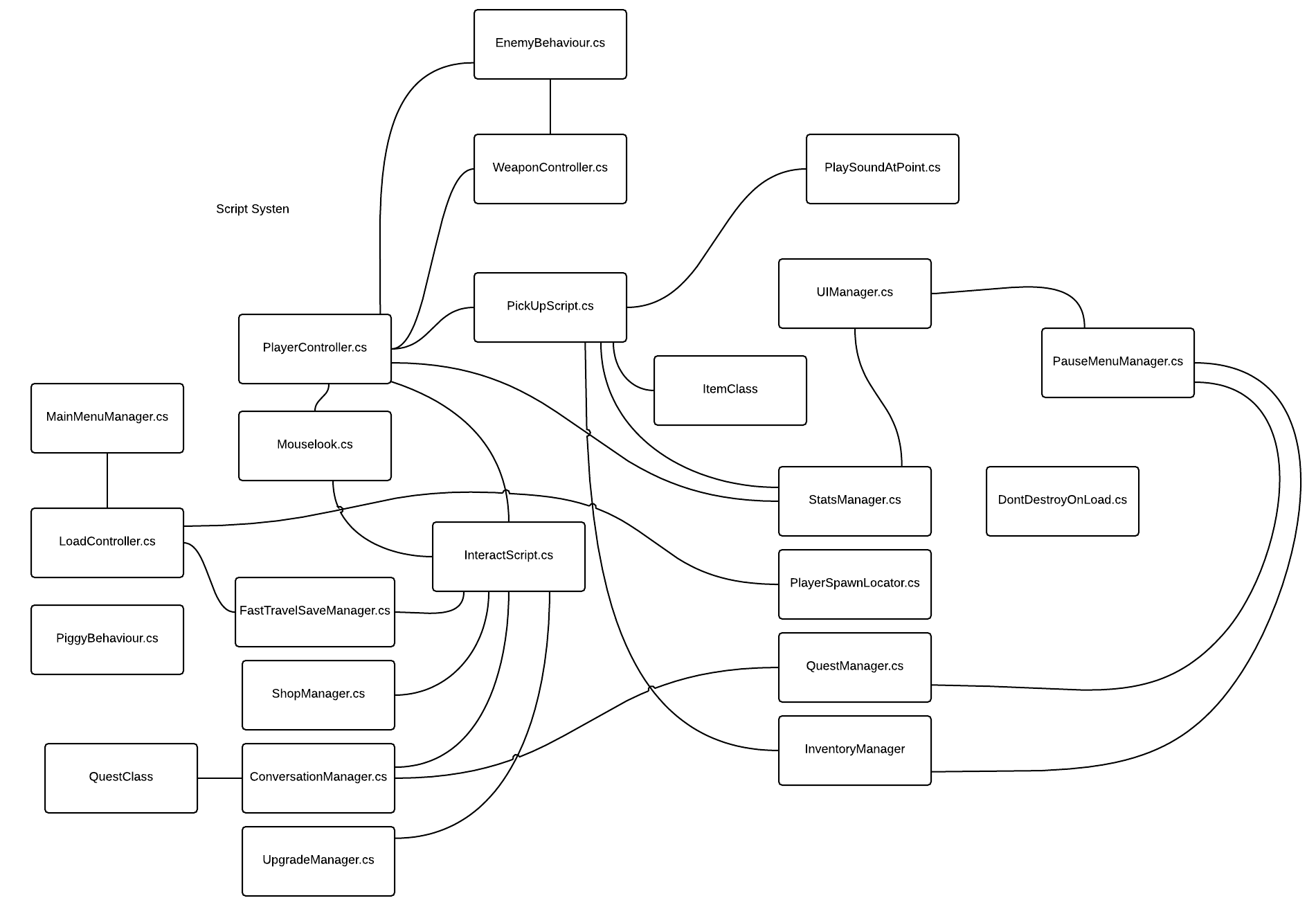
Use tabs correctly

Language structure example

RigidBody \_rb = GetComponent<RigidBody>();  
public float jumpForce;

//Jump in the air  
public void Jump()  
{  
 \_rb.AddForce(new Vector3(0, jumpForce, 0);

}

spinneweb

**Scriptbeschrijving TL;DR version**

***Playercontroller.cs***

The controls of the player will be coded here. As well as collisions of the player with other object.  
EXCEPT for Attack, that goes to the weaponcontroller.

This script also raycasts for interactable objects, if hit then pop up a text “RMB to Interact” or something, then on RMB activate the object’s InteractScript

Player controls

-Walking: WASD or Arrow keys

-Space : Jump

-Shift(hold) : Run

-RMB: Interaction

-Scrollwheel : Switching weapons

-1 t/m 5 Switch weapons (prefixed slots)

- "I" for opening inventory

- "M" for opening map

- "J " for journal / quests

- "P " for player stats

- "Esc" for menu (open & close)

***Mouselook.cs***

Script on the main camera to move around with your mouse in first person

***WeaponController.cs***

LMB to attack, each weapon has its weaponcontroller to control the animations, attack damage and stats as well as upgrades

**EnemySpawner.cs**

Instantiates an enemy as child, if it died set a timer and then respawn it.

**EnemyBehaviour.cs**

AI for the enemy. Follows a path, checks distance with player with a trigger and if it’s aggressive then attack the player. Inherits values from an enemy class

**PiggyBehaviour.cs**

Moves around and oinks in his pen

**PickUpScript.cs**

On pickup, add an item to inventory manager by calling an item class, then destroy itself and create PlaySoundAtPoint object at its location for sound (can’t play sound on a destroyed object)

**InteractScript.cs**

If interacted with, just set a Boolean on true. ConversationManager, ShopManager, UpgradeManager and FastTravelSafeManager depens on this script and will activate if this Boolean is set to true

**ConversationManager.cs**

Activates the speech bubble UI objects. Generates text and send them to the UIManager.

**ShopManager.cs**

Open the BUY or SELL window. Get information from ItemClass for price. Add the item to the inventory when purchased for Piggies.

**UpgradeManager.cs**

Open the BUY or SELL window. Get information from ItemClass for price. Add the item to the inventory when purchased for Piggies.

**FastTravelSaveManager.cs**

Open a menu for Saving or fast travel. On save put the stats from statsmanager to playerprefs. On fasttravel, set the PlayerSpawnLocator and (re)load the scene

**StatsManager.cs**

Keep track of all stats of player. Just that.

**QuestsManager.cs**

Keep track of the quests. PauseMenuManager gets info about the quest’s state and name from this script to display.

**PlayerSpawnLocator.cs**

Set the location of the player when he dies or when he fast-travels.

**InventoryManager.cs**

Keeps track of the items that are currently in the inventory, takes them from ItemClass.  
Also keeps track of weapons collected and piggies collected.

*The UI is devided into 4 scripts.*

**UIManager.cs**

Handles all UI on-screen except for the playermenu (pause, inventory, quests, map) and main menu.

**PauseMenuManager.cs**

Handles UI in the PauseMenu.

**MainMenuManager.cs**

Handles the UI in the Main Menu

**CreditsManager.cs**

Handles scrolling of the text in the credits scene.

**LoadController.cs**

This is an extra script that will show a loading screen. While this is extra, the script has already been made for an earlier project, we can just implement this in the first version of the game.

**PlaySoundAtPoint.cs**

An object with this script is required if we want an destroyed object to play a sound. Think of Coin Pickup sound or Enemy defeat sound.

**DontDestroyAtLoad.cs**

The GameManager GameObject will have this script as well as Inventory Manager, StatsManager and QuestManager.

**Classes**

**Questclass.cs**

Class and Constructors for all kinds of quests and their rewards

**ItemClass.cs**

Class and Constructors for all kinds of items and their stats, useableness, selling price and buying price etc.

**EnemyClass.cs**

Class and Constructors for all kinds of enemies and their stats like hp, movespeed, attack.