COOP Action Game Kit

Documentation

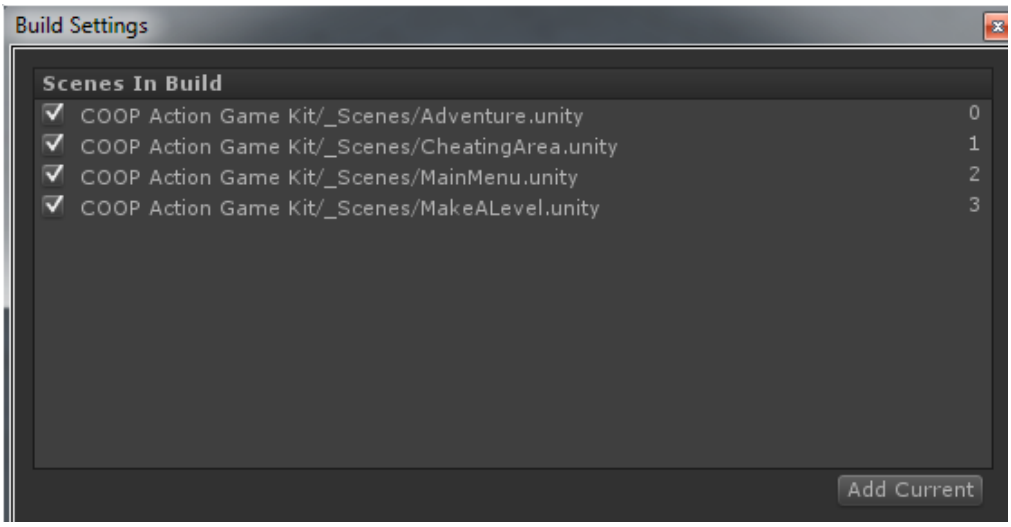
To see a up-to-date version of the documentation head over to: **http://tinyurl.com/cagkdocumentation**

Content

1. Setup
2. Replacing Art
3. Player Character
4. Enemy Character
5. Weapons
6. Projectiles
7. Spawning & Despawning Methods
8. Collectables
9. Breakable Objects
10. Adding Mobile Controls w/ Free Mobile CNJoystick
11. Procedural Level Generator
12. Sound Manager
13. Creating a Level

1. Setup

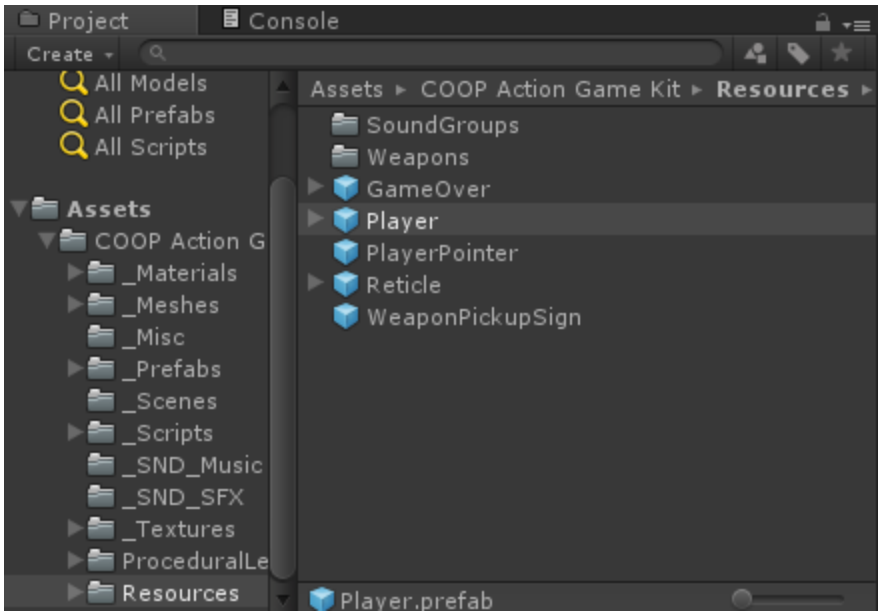
There are 2 important steps on setting up the COOP Action Game Kit.
1. First extract the contents of *"/Assets/COOP Action Game Kit/ProjectSettings.zip"* to the root of your Project, replacing the original Project Settings you had in your project. That's important so the kit can receive the input and layers correctly.
2. Include the Scenes that are inside *"Assets/COOP Action Game Kit/_Scenes"* in your Build settings, otherwise the game will be unable to load the scenes and menu.



2. Replacing Art

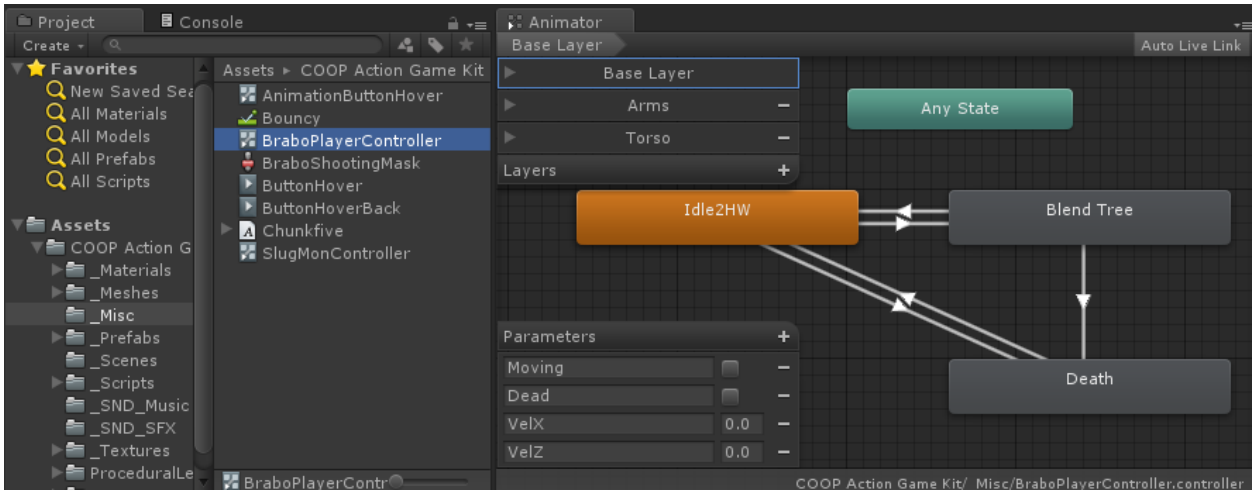
Replacing the Player Art

1. Drag the Player Prefab into a scene. The Default Player Prefab is found on “/Assets/Coop Action Game Kit/Resources/” and it’s called “Player”.



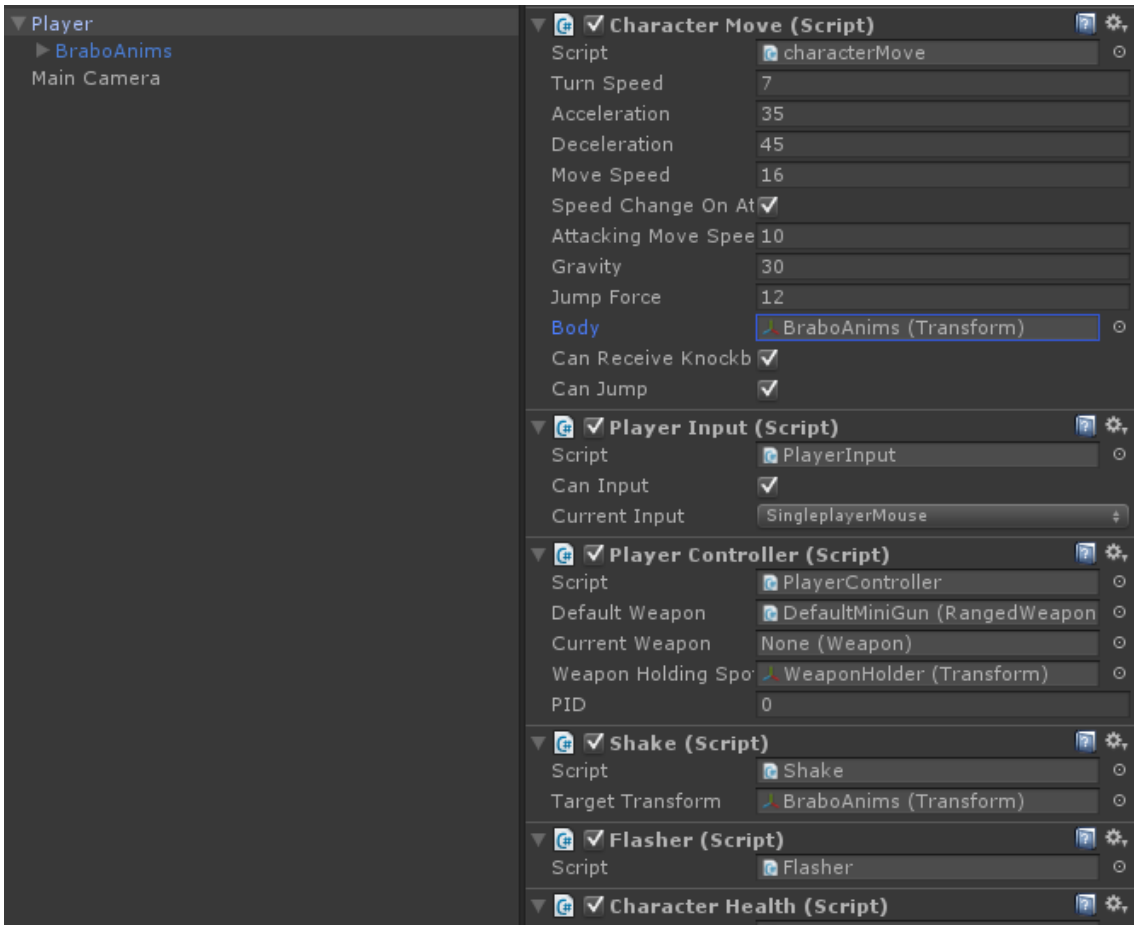
2. Inside the Player GameObject there is a GameObject called “BraboAnims”, that’s the Art for the Player, Delete it and add inside of the Player GameObject the art you want to use for your player.

3. In the folder “/Assets/Coop Action Game Kit/_Misc” you’ll find an Animation Controller called “Brabo Player Controller”. Open it and link the animations of your new character art into each animation state. Afterwards connect it into the Animator component of your new Character Art.

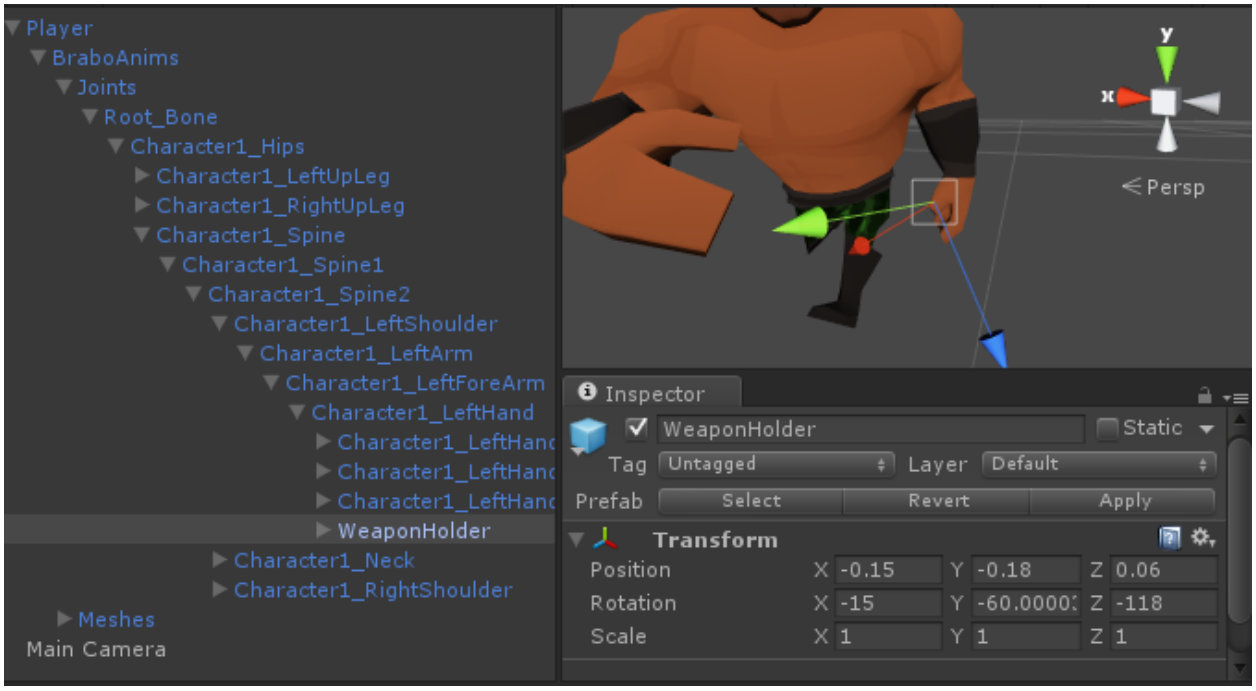


4. Keep in mind you'll need to create masks for the arms shooting animation so it'll only affect the arms and for the jump animations so it won't affect the arms. If you need to know more about that watch the video tutorial on how to Replace Art.

5. Connect your Character Visual object to: CharacterMove -> Body; Shake -> Target Transform;



6. You need to create an object for your weapon to attach. Create an empty Game Object and place it in the hiearchy under the Hand Joint of your character, and rotate it accordingly. You need to connect that to the Player Controller script in the Player GameObject, on the variable Weapon Holding Spot.



Replacing the Enemy Art

1. Drag one of the enemy Prefabs into a scene. They can be found inside "/Assets/Coop Action Game Kit/Prefabs/Enemies/".

2. Inside the EnemyGameObject there is a GameObject with the art for the Enemy, Delete it and add the art you want to use for your enemy.

3. In the folder "/Assets/Coop Action Game Kit/_Misc" you'll find an Animation Controller called "SlugMonController". Create a new copy of it and link the animations of your new enemy art into each animation state. Afterwards connect it into the Animator component of your new Enemy Art.

4. Connect your Character Visual object to: CharacterMove -> Body; Shake -> Target Transform;

3. Player Character

When spawning players, the GameManager will look for a prefab called Player inside the Resources folder. This spawn happens on GameManager.cs line 117.
The Player Character is a GameObject with the following Components:

Character Controller

Unity's Character Controller, responsible for the movement of the player.

Character Move

Component that interfaces with the Character Controller to move the character, it also has functions to control the rotation of the player body, animations related to the movement of the player and to receive knockback from other characters.

Variables

Turn Speed - Controls how fast the character rotates *(Note that when shooting this value will be overriden so that the character can reach it's desired shooting rotation faster)*

Acceleration - This value controls how quick the player will increase it's speed.

Deceleration - This value controls how quick the player comes to a stop when it stops being accelerated.

Move Speed - This value controls the speed of the player.

Speed Change On Attack - True/False which will define if the character should move slower while attacking.

Attacking Move Speed - This value controls the speed of the player while he is attacking, if "Speed Change on Attack" is turned on.

Gravity - This value controls how quick the player is pulled back to the ground while on the air.

Jump Force - This value controls the ammount of force the player receives when pressing the jump button.

Body - This Variable needs a reference to the actual Body it's controlling *(The visual representation of the character that should be inside the Player Game Object)*

Can Receive Knockback - True/False which will define wether the character can be knocked back by other attacks and external forces or not.

Can Jump - True/False which will define wether the player can Jump.

Player Input

Component that receives input from the player and interfaces with the other components to use said input.

Player Controller

Main Player Component, keeps references to the other major components in the player and helps them interface with each other. This is also the component that enemies use to find and attack the player. It also manages the player current weapon and has functions to equip and unequip weapons.

Variables

Default Weapon - This Variable needs to hold a reference to a prefab for the default weapon that the player should equip on spawn and always keep (Undroppable).

Current Weapon - This Variable will hold a reference to the current weapon being used

Weapon Holding Spot - Reference to the place where weapons should be in the hiearchy. Usually use a Empty Game Object placed and rotated correcly in the hierarchy, in the place you want your weapons to appear.

PID - This value is the ID of the player, it's set by the Game Manager when spawning players, starting on 0.

Character Health

This component controls the health of the character. It has functions to take damage, heal damage, die and ressurrect. It sends a Message OnHeal to the other scripts when the character receives a heal, OnDamage when it receives a damage, and OnDeath when it dies. Those messages can be used to trigger specific behavior on other components.

Variables

Max Health- This variable is the starting health of the character. When using the Heal method it'll cap the health at this max ammount.

Current Health- This variable is set to the Max Health value once the game start, I only left it exposed for debugging purposes and to be able to heal myself while testing.

Death Effect - This is the prefab to be spawned when the character dies (OnDeath();). Can be used to spawn a Particle Effect for the player death.

Death Sound Effect- This variable holds a string for the Sound Manager to find and play the SoundGroupManager with the sounds of Player Death. If in doubt, read more about the SoundManager.

Hit Effect - This is the prefab to be spawned when the character takes damage (OnDamage();).

Hit Sound Effect- This variable holds a string with the name of the SoundGroupManager that should be played when the character takes damage.

Heal Effect - This is the prefab to be spawned when the character takes damage (OnHeal();).

Flash Amount - This is the amount of flash the character should receive on TakeDamage by default (If the attack is not specifying it on it's attack function, it usually specifies it though.)

Shake Amount - This is the amount of shake the character should receive on TakeDamage by default (If the attack is not specifying it on it's attack function, it usually specifies it though.)

Should Respawn - True/False value that controls whether the character should respawn once it dies.

Recycle on Death - True/False value that controls whether the character should go back to the pool on Death, being "recycled".

Hurt Interval - This is the amount of time the character should wait before being able to be hurt again (How long does he stays "invulnerable" after being hit).

Shake

This component is responsible for shaking the character on damage. If you don't want the character to shake when taking damage remove this component from the character.

Variables

Target Transform - The object that should be shaken, in this case the gameobject that's the visual representation of the player.

Flasher

This component is responsible for flashing the character on damage. If you don’t want the character to flash when taking damage remove this component from the character.

Player Health Bar Controller

This component looks for a Health Bar Manager in the scene and will interface with it, giving information on the current health on damage.

4. Enemy Character

Enemy Characters are gameobjects that contain another gameobject with a visual representation of the enemy, very similar to the player character. It also shares some of the components that the Player Character have. It usually contains:

Character Controller
Unity's Character Controller, responsible for the movement of the player.

Character Health
This component controls the health of the character. It has functions to take damage, heal damage, die and ressurrect. It sends a Message OnHeal to the other scripts when the character receives a heal, OnDamage when it receives a damage, and OnDeath when it dies. Those messages can be used to trigger specific behavior on other components. To know more check the description on "3. Player Character"

Shake
This component is responsible for shaking the character on damage. If you don't want the character to shake when taking damage remove this component from the character.

Flasher
This component is responsible for flashing the character on damage. If you don't want the character to flash when taking damage remove this component from the character.

Enemy AI
This is the component that regulates the Enemy's Artificial Intelligence. It's governed by several variables and can be extended to change it's behavior.

Variables:

Player Detection Settings

Detection Range - This is the range in which the enemy will be looking for players. If you want the enemy to find the player anywhere in the map just increase this value to a really high amount.

Detection Interval - This is the interval in which it's actively looking for players. So it doesn't look for it every frame. It's also used to know if we should check if there is a better player to attack (closer) in some of the other states.

Attack Settings

Enemy Attack - This is the EnemyAttack component that this attacker will be using. You need to plug in here an object with a EnemyAttack component (Or one of

it's children). It doesn't need to be the same gameobject, but it can be. On my examples the atack component is in the same gameobject.

Attack At Angle - The enemy will only attack if the player he's attacking is inside this angle in relation to the enemy. If you want him to attack independent of the angle between player/enemy increase this value to 360.

Attack At Min Range - This is the minimum range for the enemy to try and attack. He'll try to get to this range of the player before effectively attacking.

Attack At Max Range - This is the maximum range for the enemy to attack, if the player is not within this max range the enemy will go back to the Chase state to try and get closer to the player.

Attack Interval - This is the interval between attacks. The enemy will wait this amount of time before attacking again.

Rotate During Attack - This TRUE/FALSE value controls whether the enemy should rotate while attacking to keep facing the player or not.

Move During Attack - This TRUE/FALSE value controls whether the enemy should keep moving towards the player while attacking or not.

Evade Settings (WIP, not ready).

Hurt Settings

Stop On Hurt - This TRUE/FALSE value controls whether the enemy should stop his action when being hurt. He'll change to the Hurt State and wait before resuming his previous action.

Recover Duration - This is the amount of time he should wait before resuming his action after being hurt if "Stop On Hurt" is turned on.

Stun Settings (WIP, not ready).

Can Roam Around - This TRUE/FALSE value controls whether the enemy should roam around if he haven't seen the player yet.

Log - This TRUE/FALSE value turns on Debug.Logs and ONGUI value showing, good for when you're programming/extending the AI and want to know in which state the AI is, and what is going on with it.

Enemy Attack Ranged

This is a component extended from EnemyAttack that is used to execute Ranged Attacks.

Variables:

Shake Intensity - The camera will shake with this amount of intensity when the attack happens.

Wait Before Attack - This is the amount of time that we'll wait before actually spawning the projectile. (If you need to wait for an animation to reach a certain point before actually doing the attack).

Wait After Attack - This is the amount of time that we'll wait after the attack before going back to the state we were in before.

Projectile Settings

Projectile Object - This is the actual prefab that will be spawned on the ranged attack. It needs to contain a "projectile" component.

Projectile Speed - This controls how fast should the projectile be.

Projectile Range - This controls how far the projectile gets before being despawned.

Projectile Angle Variation - This controls the variation on the angle of the projectile. So the shots don't go all in the same exact direction. If you have more than 1 bullet per shot, the first bullet will go exactly on the angle the player asked for and the others will have variation applied. So for a shotgun-like weapon it's nice to get a high number on this variable.

Shoot Settings

Muzzle Effect - This is a reference to the prefab that should be spawned at the Spawn Point when a shot happens. Usually a Muzzle FX.

Spawn Point - This is the Spawn Point where the projectiles should spawn at. You should create an Empty Game Object and position it where you want the projectiles to be spawned, link that game object in this variable.

Shot Sound Effect - This variable needs a string that it'll pass to the SoundManager for it to play a sound on every shot.

Bullets Per Shot - This controls the amount of bullets that's spawned on each attack.

Bullets Per Shot Interval - This controls the interval between the bullets spawned in the shot (If there is more than 1 bullet being spawned per shot).

Weapon Kick - This controls the knockback force the character receives from shooting.

Weapon Kick Duration - This controls how long the knockback force from shooting is applied on the character.

Shoot Animation String - This is the String for the Animation State that should be played on Mecanim for this attack.

Enemy Attack Melee

This is a component extended from EnemyAttack that is used to execute Melee Attacks.

Variables:

Shake Intensity - The camera will shake with this amount of intensity when the attack happens.

Wait Before Attack - This is the amount of time that we'll wait before actually starting to deal damage. (If you need to wait for an animation to reach a certain point before actually doing the attack).

Wait After Attack - This is the amount of time that we'll wait after the attack before going back to the state we were in before.

Melee Settings

Attack Radius - This controls the radius in which the damage will be applied.

Attack Damage - This controls the amount of damage that will be applied to the characters in range.

Attack Duration - This controls how long the attack will be happening.

Attack Angle - This is the angle in which the attack will be applied. If you want the attack to happen all around the character input 360 there. If you only want it to happen in front of the character input around 65~70 there.

Attack Effect - This holds a reference to a Prefab that should be spawned on attack.

Hit Effect - This holds a reference to a Prefab that should be spawned when the attack hits something.

Attack Animation - This is the String for the Animation State that should be played on Mecanim for this attack.

Knockback Settings

Knockback Force - This controls the Knockback Force that should be applied to the characters hit by this attack.

Knockback Duration - This controls the duration in which the Knockback Force should be applied to the characters hit by this attack.

Affected Actor Mask - This controls which layers are affected by this attack.

Enemy Attack Self Destruct

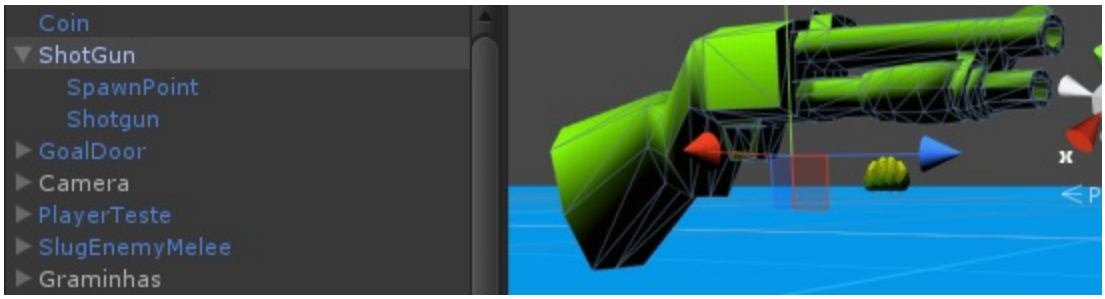
This is a component extended from EnemyAttack that is used to Self Destruct.

Variables:

Shake Intensity - The camera will shake with this amount of intensity when the Self Destruct happens.

5. Weapons

Weapons are GameObjects that contain a Component that extends from the Weapon Component. Inside this gameObject there should be a gameobject with a visual representation of the weapon and in the case of a ranged weapon, a Spawn Point GameObject, positioned where the projectiles should spawn. The Pivot of the Weapon GameObject should be positioned where the Weapon should be pivoting at. In the case of weapons that would usually be the handle.



It's important to save your Weapon Prefabs inside "Resources\Weapons\". To spawn the weapons when passing through levels the GameManager will look for a prefab of the weapon in "Resources\Weapons\" by it's name, so if it can't find one you'll lose the weapon you're holding when passing from one level to another.

Ranged Weapon

This is a component extended from Weapon that is used to execute Ranged Attacks.

Variables:

Attack Cooldown - The amount of time after attacking until the player is allowed to attack again.

Shake Intensity - The amount of shake that should happen to the camera when the weapon fires.

Projectile Settings

Projectile Object - This is the actual prefab that will be spawned on the ranged attack. It needs to contain a "projectile" component.

Projectile Speed - This controls how fast should the projectile be.

Projectile Range - This controls how far the projectile gets before being despawned.

Projectile Angle Variation - This controls the variation on the angle of the projectile. So the shots don't go all in the same exact direction. If you have more than 1 bullet per shot, the first bullet will go exactly on the angle the player asked for and the others will have variation applied. So for a shotgun-like weapon it's nice to get a high number on this variable.

Shoot Settings

Muzzle Effect - This is a reference to the prefab that should be spawned at the Spawn Point when a shot happens. Usually a Muzzle FX.

Spawn Point - This is the Spawn Point where the projectiles should spawn at. You should create an Empty Game Object and position it where you want the projectiles to be spawned, link that game object in this variable.

Shot Sound Effect - This variable needs a string that it'll pass to the SoundManager for it to play a sound on every shot.

Bullets Per Shot - This controls the amount of bullets that's spawned on each attack.

Bullets Per Shot Interval - This controls the interval between the bullets spawned in the shot (If there is more than 1 bullet being spawned per shot).

Weapon Kick - This controls the knockback force the character receives from shooting.

Weapon Kick Duration - This controls how long the knockback force from shooting is applied on the character.

6. Projectiles

Projectiles are GameObjects that contain a “Projectile” component and usually another GameObject inside of it with a visual representation of the projectile.

Variables:

Hit Effect - This is a reference to a Prefab that should be spawned on hit, usually a Impact Particle Effect.

Hit Sound - The String that should be sent to the SoundManager for playing a sound when the projectile hits something.

Trail Effect - This is a reference to a prefab with the Component **TrailEffect**. This component handles following the Projectile that spawned it and will self-despawn when it’s not needed, making sure that the particle stops emitting so when it gets out of the pool it doesn’t emit trail where it shouldn’t.

Radius - The radius of the projectile, this is how far from the center of the object it’ll look for things to actually hit.

Damage - This is the damage this projectile inflicts on whatever he hits.

Force - This is the force (knockback) the damage will try to inflict in what he hit.

Force Duration - This is the duration that the force will be applied to what this projectile hit.

Hit Cam Shake - This is how much camera shake should happen when the projectile hits.

Character Mask - This is the Layer Mask that the projectile should be looking for. If this is a player projectile you probably want it to hit at least the Enemy Layer. If it’s an enemy projectile you want it to hit the Player Layer.

Wall Mask - This is the Layer Mask in which your walls and decorations that shouldn’t suffer damage but should despawn the projectile are at. So the projectile will react as he hit something, but won’t try to apply damage to it.

7. Spawning & Despawning Methods

There are several components to assist Spawning and Despawning objects in the kit. I'll explain each of them here.

Spawning Objects

SpawnOnDeath & SpawnOnStart

These components can be used to spawn prefabs when the object starts on when the object receives "OnDeath" message. They share these variables:

SpawnObjects - This is an array to place the objects you want to be spawned. They can be spawned at random or spawned all of them in order.

Spawn Random Quantity Min - The minimum amount of objects that should be spawned if using "Spawn Random".

Spawn Random Quantity Max - The maximum amount of objects (exclusive) that should be spawned if using "Spawn Random".

Spawn Random - True/False value controlling whether it should spawn random objects from the list using the min/max quantity or to spawn all the objects in the list in order.

Position Variation Factor - The variation in the position when spawning several objects. If set to 0 it will spawn all the objects in the same place.

WeightedLootSpawner

This component can be used to spawn prefabs based on a set weight. Higher the weight, higher the chance the object will be spawned.

Items - This is an array in which each element holds a few variables. The variables are these:

Prefab - This is the prefab that will be spawned by this component.

Weight - This is the weight of this item, the probability that this will spawn.

Spawn Y - This is a value of Y to be added on top of the Spawner Object transform position. It's useful if you have your weighted spawner on the floor but you want your object to be spawned above the floor a bit.

Spawn on Death - True/False value that controls whether this spawner should be activated on Death or not.

Spawn on Start - True/False value that controls whether this spawner should be activated on Start or not.

Spawn Random Quantity Min - The minimum amount of objects that should be spawned if using “Spawn Random”.

Spawn Random Quantity Max - The maximum amount of objects (exclusive) that should be spawned if using “Spawn Random”.

Spawn Random - True/False value controlling whether it should spawn random objects from the list using the min/max quantity or to spawn all the objects in the list in order.

Position Variation Factor - The variation in the position when spawning several objects. If set to 0 it will spawn all the objects in the same place.

Despawning Objects

TimedDespawn

Putting this component into an object will despawn the object after the set amount of time. You set the time in the variable TimeToDespawn. It’s useful to use in Particles so they’ll despawn after playing.

DespawnOnDeath

Putting this component into an object will despawn the object when it receives a OnDeath message, after waiting the set amount of time. You set the time in the variable TimeToDespawn.

Executing Actions on Spawn

ApplyForceOnSpawn

This can be used to apply a random force to the spawned objects, useful for the coins and powerups if we want them to “explode” out of the enemy/crate.

Apply Random Drag - True/False value that controls whether a random drag value should be applied to the rigidbodies in the object.

Random Drag Min/Random Drag Max - Minimum/Maximum value of the random drag, if you chose to apply it.

ApplyTorqueY - True/False value that controls whether a random Torque Y force should be applied to the object.

RandomTorqueYForce - A multiplier for the Torque Y Force that should be applied to the object if you chose to apply it.

UpRandomForce - A multiplier for the Up Y Force that should be applied to the object. (0 to apply no force on the Y axis.)

RandomForce - A multiplier for the Force that should be applied to the object.

DamageOnSpawn

This is basically a melee attack, but it happens when the object is spawned. Look into the variables of the enemy melee attack if you want to know more about the variables in here. It can also be used to constantly give damage at that point at a set interval. Good to put “traps” on the set. The variables that differ from the enemy Melee attack are:

DestroyAfterwards- True/False whether we should despawn the object after the attack happens.

Repeating - True/False whether the attack should try and repeat after the attack happens.

AttackCooldown - If the attack should repeat, this variable controls how long to wait before performing the attack again.

AttackDuration- This value sets how long the attack should be applying damage to whoever is in range.

PickRandomMaterial

This component will take Materials from the PickFromMaterial array and randomly assign one of them to the object. Useful to apply extra variation for the objects being spawned out of the Procedural Level Generator.

ScaleUpOnSpawn

This component will scale up the object from Vector3.zero to it’s original value on spawn. you can control the duration and the amount of time it should wait before scaling on the variables.

RadomRotationOnSpawn

This component will randomly rotate the object in the Y axis on spawn. It has no variables.

PositionOnFloor

This component will raycast down looking for colllision with a layer specified in the “FloorMask” variable. When it finds that point it’ll pick it’s Y position and add the position set in the variable “WantedFloorDistance” to it, setting the resulting value to the Y Position of the object.

TLDR: It will place the object above the floor.

Other

TrailEffect

This component is used on the trail particles for the projectiles. The projectile spawns a trail object that contains this component and tells this component to follow it. Once it dies it tells this component to despawn itself and reset it’s particle system so it won’t show up until it’s respawned.

8. Collectables

PickupObject

This is the base component responsible for pickups in the game. By default he is used to affect player's score on pickup, but can be extended by overriding "PickupEffect(PlayerController);". It needs a trigger for it to check for the distance for pickup and to apply the magnet force once the player is inside it. If there is no trigger it'll create one for you. Also if there's no rigidbody on the object it'll create one on spawn and set it to kinematic.

Variables

ScoreValue - The amount of score the player gets when they pick this object.

PickupFX - The prefab that should be spawned when the object is picked up.

PickupSound - This contains a string that is sent to the SoundManager for it to play when the object is picked up.

MagnetForce - This is the force that should be applied towards the player while the player is inside the trigger.

DistanceToPickUp - This is the distance that the object needs to be from the player for it to be consumed/Picked Up.

CantPickupOnSpawn - Amount of time after spawning in which it's not possible to pickup the object.

PickupHeal

This is an extension of PickupObject, and it overrides "PickupEffect()" to heal the player instead of adding score to the player.

Variables

HealAmount - The amount of health points to heal when picked up.

OptionalHealPercentage - Percentage to heal (0 to 1, so if I want to heal 50% I'd put 0.5 here).

ShouldHealPercentage - True/False value which controls whether to heal using OptionalHealPercentage or HealAmount.

9. Breakable Objects

Breakable Objects are simply objects with a **CharacterHealth** component with the RecycleOnDeath option turned on. Now, if you want it to be able to receive knockback you need to include a **Unity's CharacterController** and a **CharacterMove** on it.
If you want it to be able to shake when receiving damage, simply add a **Shake** Component to it, and if you want it to flash when receiving damage, add a **Flasher** component to it.

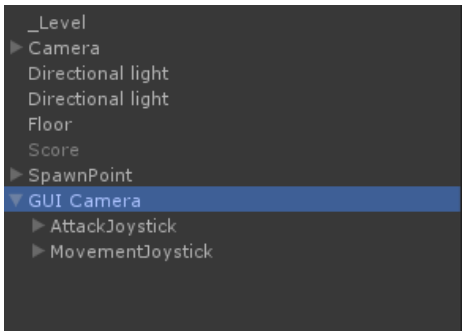
Now, for the drops, you can use some of the **Spawn Methods** components.

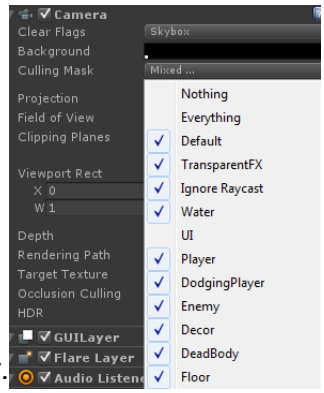
Use Spawn On Death to spawn destroyed pieces of the breakable objects, or to spawn Coins and other collectables. Or to spawn weapons with a low percentage of chance, use the Weighted Loot Spawner.

10. Adding Mobile Controls w/ Free Mobile CNJoystick

Download and import Mobile CNJoystick from the following link:
https://www.assetstore.unity3d.com/en/#!/content/15233

Copy the GUI Camera object from the *"CNJoystick\DemoScene"* over to the scenes where the game will happen. (In this demo case, the "Adventure" scene.) Set the GUI Camera's layer to **UI** and all of it's children as well.





Make sure your regular camera (Camera) is set to not show the UI layer.

Set one of the Joysticks to the Tag *"AttackJoystick"* and another to *"MovementJoystick"*. If you want to test it using the mouse you need to uncheck the checkbox *"Remote Testing"* on them.

Drop the **PlayerInputMobile** component into the Player Character prefab (Usually located in the Resources folder. Open *PlayerInputMobile.cs* and delete "/*" from line number 6 and "*/" from line 106. You should be good to go now.

11. Procedural Level Generator

The Procedural Level Generator uses the Celular Automata Method, it's based off of this article on the Roguebasin wiki. If you want to know more about how it works, I suggest you read that article.

Now, this level generator is one of the first pieces of code I created when learning Unity, before I even knew how to move a character around. So It's a bit of a mess. I do intend on improving it in the future, but for now it's usable, but won't be really fast to generate levels, and will definetly not generate something optimized for mobiles, for my demo levels I'm getting around 3500 drawcalls. Having said that, it still produces some nice levels for PC games, and you can get the levels it generates, CTRL+C and CTRL+V them into a new scene and optimize them yourself, by combining meshes and materials and stuff.

Alright, so here are the variables and what they actually do.

Variables

MAP_WIDTH - This is the Width of the desired Map.

MAP_HEIGHT - This is the Height of the desired Map.

WALL_PERCENTAGE - This value controls the amount of walls that is generated on the first pass of the generator. It shouldn't go much higher than **50** otherwise levels will be really really small. If you go way too much down from **45**, levels will be very very open.

Floor - If you plug a gameobject here (A plane for the floor). It'll adjust it's size, position it and try to tile it's material accordingly.

Use Seed - This TRUE/FALSE value will control whether the generator should use the Seed number or just pick a random number for the level.

Seed - This value is used to generate the map if USE SEED is turned On. However note that if the seed in question is incapable of generating a good result (A result with a Goal Object that's distant enough from the player) it'll discard the seed after 10 tries and generate a new one.

Spawn Players - TRUE/FALSE value whether players should be spawned by the GameManager or not when the level finishes generating.

Objects to Spawn

Filled Space Array - The Generator picks a random object from this array for the spaces that should be filled outside the direct walls of the map.

Straight Wall Array - The generator picks a random object from this array for the straight walls.

Curved Inside Wall Array - The generator picks a random object from this array to connect walls from the inside (on wall curves).

Curved Outside Wall - The generator picks a random object from this array to connect walls from the outside (on wall curves).

Column Wall - The generator picks a random object from this array when it doesn't know what to spawn for the combination of wall connections.

Decor Array - The generator picks a random object from this array to spawn in places it judges good for decoration based on the variables of decoration distance from one another.

Breakable Objects Array - The generator picks a random object from this array to spawn in places it judges good for breakable objects based on the variables of breakable objs distance from one another.

Enemy Spawners Array - The generator picks a random object from this array to spawn in places it judges good for enemies based on the variables of decoration distance from one another.

Treasure Array - The generator picks a random object from this array to spawn in places it judges good for treasures based on the variables of treasure distance from one another.

Goal Level - This is the prefab it's going to spawn as far from the player as possible in a corner to be the goal of the player.

Spawn Point - This object will not be spawend, it should be linked to an object already in the scene and will be moved to the place where the player should be. So either link a Player here, or a SpawnPoint for the players using the SpawnPoint Prefab or Component.

Tile Size - This is a multiplier for the grid when spawning the objects, it should be the size of the tiles, in my case, I'm using 3x3 tiles on the demo.

Enemy Spawner Distance - This is the minimum distance on which to spawn enemy spawners. A bigger number will mean less Enemy Spawners, while a smaller number means more enemy spawners.

Decoration Spawner Distance - This is the minimum distance on which to spawn decoration spawners. A bigger number will mean less Decoration Spawners, while a smaller number means more decoration spawners.

Breakable Obj Spawner Distance - This is the minimum distance on which to spawn Breakable Obj spawners. A bigger number will mean less Breakable Obj Spawners, while a smaller number means more Breakable Obj spawners.

Treasure Obj Spawner Distance - This is the minimum distance on which to spawn treasure spawners. A bigger number will mean less treasure Spawners, while a smaller number means more treasure spawners.

12. Sound Manager

The Sound Manager is divided between the actual Singleton that controls the sound **“SoundManager”** and a smaller Component called **“SoundGroupManager”**. SoundGroupManager should be in a prefab inside *“Resources\SoundGroups”* with the name you’re calling Sound Manager to play. SoundManager will on start cache a reference to all the SoundGroupManager prefabs inside *“Resources\SoundGroups”* . And when it receives a Play call with a string, it’ll look on it’s dictionary for the corresponding SoundGroupManager, spawning a new one if there isn’t one already in the pool, or reutilizing a used one from the pool.

SoundGroupManager prefabs will automatically go back to the pool once it stops playing.

SoundManager also looks for a SoundGroupManager prefab on *“Resources\SoundGroups”* with the name of the Level you’re currently playing, and will play that group on spawn. If you check the TRUE/FALSE value “Music” on the group, it’ll continue to play in order instead of despawning after finishing playing the audio.

I suggest you use 2D Audio instead of 3D audio for this kit, since the camera by default is so far away from the player, unless you put an Audio Listener on the player itself. The SoundManager can receive a position Vector3 to play the sound at.

SoundManager

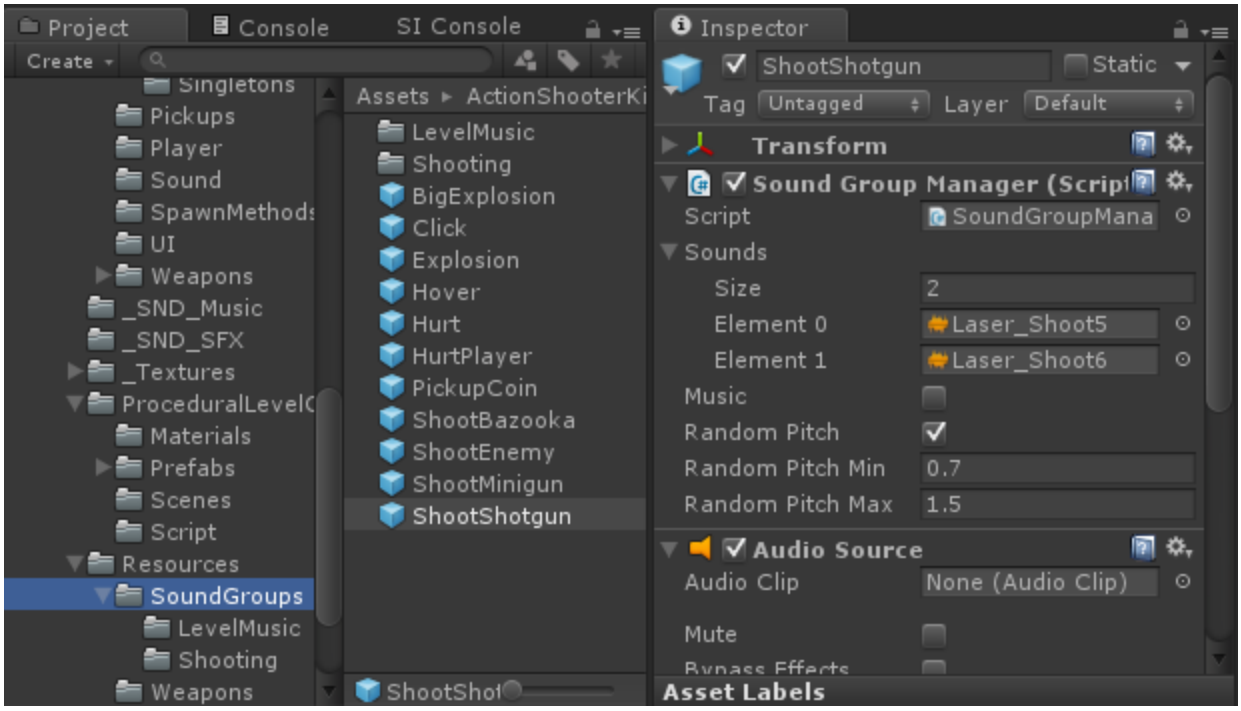
This singleton gets spawned when someone calls for him using “SoundManager.instance.function or variable”. It has a SFXVolume variable that can be changed to change the volume of sound effects. And a MusicVolume variable that can be used to change the volume of musics.

You can call it to play a sound by calling from any script: *“SoundManager.instance.Play(INSERT SOUNDGROUPMANAGER PREFAB STRING NAME HERE, INSERT VECTOR 3 POSITION HERE)”.* Or a music by calling: *“SoundManager.instance.PlayMusic(Insert SoundGroupManager Music String name here”.*

If you want to play it at 0,0,0 just call *SoundManager.instance.Play(SoundManagerGroup Name);.*

SoundGroupManager

To get your Sound Effects playing and pooling, you need to create prefabs with the Sound Group Manager component inside of *"Resources\SoundGroups"* It has an array called "Sounds" and it'll pick automatically a sound from that array when it's spawned and play it. If you mark the "Random Pitch" box, it'll pick a random value for the Pitch between "Random Pitch Min" and "Random Pitch Max" on each play. The name of the prefab should be the name that's being called on the other scripts, so for example in the weapon prefab I would have that exact name "ShootShotgun" in the "Shot Sound Effects" variable.



13. Creating a Level

To create a level first delete the camera you got in your new level and pickup a Camera prefab under the “Resources” folder. The Camera prefab comes with the CamManager component, that follows the players around.

Afterwards if you want your players to spawn at a specific point, position a SpawnPoint Prefab (You can find one in the “Resources” Folder) on the scene, otherwise your players will be spawned at Vector3.zero. Make sure the SpawnPoint is effectively above the ground otherwise your player will fall through it.

If you want your level to have music. Make sure you create a new GameObject with the SoundGroupManager component, check “Music” and include your audio there. Save a prefab of it to the “Resources\SoundGroups”, it must have the SAME NAME as your level, if it does, SoundManager will play it automatically once you start the level.

If you want to place a Portal in your level that will take you to a next level or something, pick up a “Goal Door” prefab in the Resources folder. In the GoalDoor Component, you can put a name of a level to be loaded when the goal is touched by the player. Make sure you include that level in your Build Settings though.

If you want to test your level without having to go through the menu, you can place a Player Prefab there directly, by dragging the “Player” prefab out of the “Resources” folder into your level.

If you want the Menu to take you to your new level instead of taking you to adventure, find the PlayButton GameObject in the Menu3 and change “StartLevelName” to the name of your level.