GAME KIT CONTROLLER 2.3.6

By Two Cubes

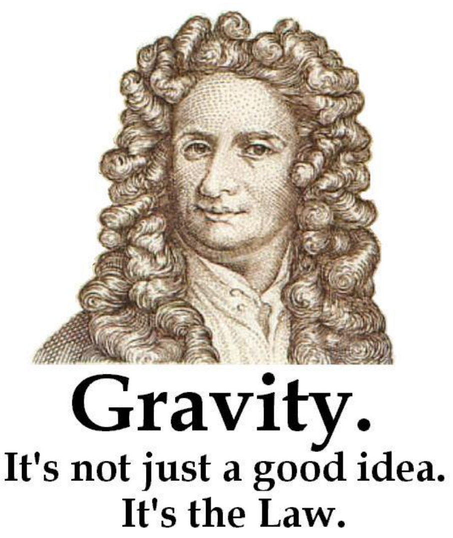


Table of content

INTRODUCTION

Optimized for Unity 4, 5 and mobile.

Solid 1st/3rd Person Controller with unique Gravity and Sci-Fi features! (+150)

Biggest update ever 2.3.6 Beta with reduced price From $40 to $35

GKC is a powerful game framework which allows you to create most type of games very easily.

It is the perfect foundation for your game and a great learning tool. There are tons of possibilities.

You can set up a character controller and test it in a few seconds.

Very customizable effects and options: camera states, configuration and transitions, power system, weapons, touch controls, headbob, custom input keys, footsteps, vehicles, etc...

WARNING: THIS DOCUMENTATION IS NOT COMPLETE FOR UPDATE 2.3.6. IT WILL BE AVALIABLE VERY SOON.

FIRST STEPS AND BASIC USE

INPUT MANAGER

You can check all the actions in the pause menu by pressing Esc, then select Edit Control Input. In that window you can see all the actions defined in the Input Manager. This component is assigned in the gameObject “Character”.

In the inspector you can add, remove, enable and disable every action and assign a keyboard button, a gamepad button and a touch button.

Also, you can edit the path to save the current map button. This file will be stored in the persistent path by default or in a relative path to your project folder if you enable the option “Use Relative Path”.

In the options below, you can select the keyboard or touch controls as input when the game starts. Also, in game you can switch between these two modes with “C” by default if you are in keyboard controls. If you are in touch controls, press Esc and then press “Switch Control Type”. This option can be used in both modes.

PLAYER CONTROL MODE

The player has 3 modes: weapons, powers and combat. To change between them, press H (by default). You can see the current state by checking the name in the upper left corner of the screen, below the health and energy bar. The weapon mode is the default mode. In Player States Manager (in the gameObject Character) you can choose the default mode at the beginning of the game.

CREATE A NEW CHARACTER

To set your model:

- In the tools bar press Game Kit Controller and Create New Player.
- In that window select a humanoid model and press the Create Player button.
- Then you will see the messages in the console about the weapons. If you want to use them, follow the next steps.

SET THE DEFAULT WEAPONS IN THE NEW MODEL

This steps are not necessary if you do not need to use the weapons:

- Go to Player Controller gameObject, and to the inspector Player Weapons Manager.
- Open Show Weapon List and press Get Weapon List.
- You can now press the Enabled option in every weapon to make it usable in game (else the weapon model in the player is disabled, but you can use it if you pick the weapon in the level, but you need first to complete these steps to avoid problems. You can remove every weapon in the list, so even if the player finds that weapon to pick in the level, he will be unable to use it)
- Select inside the gameObject Weapons (inside the player), all the weapons, open the inspector IKWeaponSystem.
- Press the button Show Element Settings and then in Show Third Person Settings. Go to hands info and assign the right and left skeleton hand of the model in every label called Hand Transform.
- Finally, you can assign in every weapon where will be attached inside the players body (the back, the legs, the waist and by default is the back), so if the weapon is in the leg, it is place inside of it. To this go to any weapon, go to gunModel inside the weapon, and open the inspector PlayerWeaponSystem. Press the button Show Weapon Settings, and set the bone that you want in the label Weapon Parent (for example in the demo, the pistols are in every leg and the rest of weapons in the back of the player).

FEATURES AND LOGS

- Walk and run in any surface
- Camera collision detection
- Move in the air to search a new surface
- Accelerate your movement in the air
- Circumnavigate spheres or regular surfaces
- Run while you adhere to any surface
- Grab and carry objects to drop or throw them
- Physics-based controller
- New menu with Unity UI
- Multi screen resolution
- Touch buttons and joysticks ready to use
- Change between first and third person view

- New first person character added without animator or mecanim needed
- Save/load and edit touch buttons positions
- Time bullet ability
- Fully commented C# code and documentation

LOG 1.5 WHAT IS NEW IN THIS VERSION

- **First person mode added**, now there are two controller, the third person which can change the camera in run time from third to first view and vice versa and the first person which has the same scripts but without the Mecanim, animator and other stuff.
- **Touch controls added**, now you can test this asset in a touch device with the new control system with touch joysticks, very customizable. You can change the control system in the editor and in run time in pc and in your touch device. The joysticks are adjusted automatically to any screen size. The touch controls can be used in editor mode with the mouse, which simulates a finger tap. Also in run time you can change between them and keyboard controls.
- **New menu and HUD with touch buttons**, using the UI of unity 4.6, to add the new options and make the interface adjusts automatically to any screen size.
- **HUD buttons can be moved in run time in pc and smartphones**, to adjust their positions as you want and **that configuration is saved in a file**, so the new positions will be charged in the next execution in the editor mode and smartphones.
- **Minor fixes and improvements.**

LOG 2.0 WHAT IS NEW IN THIS VERSION

- Like 3rd shooter aim mode
- Shoot energy, grenades, black holes, push objects, slow down the movement of objects and enemies and joint objects
- Close combat system with kicks and punches
- Change left or right side to aim
- Change between powers with the mouse wheel and the numeric keyboard
- Grab objects in aim mode and drop, throw or change their gravity
- Protect from enemies with a shield and catch and throw enemies shoots with it
- Displace objects on rails
- Deflect and project laser with different colors, and use them as weapons
- Connect different devices using lasers
- Improved HUD with regenerative health and power system
- Signaling targets with in screen icon
- Easy customizable powers system
- Particles added to every power

- Ragdoll to mecanim and vice versa
- Enemies and objectives radar system
- AI Enemy turrets with different weapons
- Hackable turrets by sneaking or slowing down to make them your allies, to grab and set them in any surface
- Quick Time Event to hack turrets and devices
- Health bars above enemies and allies
- Arrow icon to show the direction of enemies attacks and screen fading for damage
- Swipe in touch devices to change between powers and to hack enemies
- Footsteps audio system with ground physics detection
- Zoom mode and move away the camera

LOG 2.3 WHAT IS NEW IN THIS VERSION

- Improved footsteps system, with mesh and terrain texture detection and random pool of sounds, using triggers and/or raycast
- Editable key controls in game and in editor mode, with the option to save/load them
- Unified keyboard and touch controls, very configurable and customizable
- Improved power system, more configurable
- Editable key numbers powers (drag and drop)
- Selectable power by rotating the mouse
- Fixed the locked cursor in unity 5
- Fixed an issue in editing touch buttons positions
- Fixed an issue with the footsteps being disabled by the ragdoll wizard
- Code terminals using UI
- Computer terminal using UI
- Text devices using UI
- Three new powers: Implosion grenades, multiple trackable shooting objectives and change general level gravity
- Use the accelerometer to help to improve the aim mode camera
- Scanner visor to know more info about an object or an enemy and database of scannable objects
- Pickable health and power objects with icons and info in screen
- Headbob system with states
- Improved laser, with reflections in other surfaces
- Fall damage
- Set in the editor if the game starts in third person view or first person
- Double jump
- Camera collision in aim mode improved
- Gravity control system very improved, now the player can move while he searchs a new surface
- Chest with pick ups
- Features manager inspector to enable and disable every feature of the asset
- Smart use of the mouse cursor

- Procedural and very customizable door system, with different ways to open them
- Option to disable the animator in first person mode
- Select if the player uses animations or ragdoll when he dies

The first person prefab from version 2.0 has been removed, because the current controller has the same features in third and first person mode, allowing also start the game in any view mode.

LOG 2.3.5 WHAT IS NEW IN THIS VERSION

- Change between joysticks and touchpad in touch controls
- IK used in arms to aim, very easy to configure
- Aim collision detection for arms
- HUD system with health, power and weapons for player and vehicles
- Procedural Ragdoll
- Improved build player, set your model in less than 40 seconds
- Holographic doors
- Physics Vehicles system very configurable, with cars and motorbikes
- Vehicles can jump, break, boost, change camera between third and first person and move away the camera
- Regenerative health and energy for vehicles
- Accurate damage detection in vehicles
- Vehicles can be grabbed by the player, dropped and thrown or change their gravity
- When player gets off from the vehicle, his gravity is the same as the vehicle
- Vehicle obstacle detection when player gets off
- Choose side to get off from vehicles
- Set if player is ejected or he dies when vehicles explode
- Weapons system for vehicles easily configurable
- 8 types of weapons added for vehicles: machine gun, cannon, laser, homing missiles, implosion grenades, double machine gun, barrel launcher, seeker missiles
- The barrel launcher describe parables in any surface and direction
- Enemy detection when player drives
- First and third view in vehicles
- Vehicles control for touch devices
- New interaction with devices and descriptive action of them
- Slopes walking improved
- IK, weapons and gravity control system for vehicles
- No animations needed to drive
- Health, energy and ammo pickups for player and vehicles
- Now, you can hack any enemy or device
- Procedural doors and elevators system
- Explosive barrels with pickups dropping
- Recharger stations for health and energy
- Security cameras with controls and zoom
- New holographic doors
- New power, change position with another object

- Sounds for most of the systems
- Lot of fixes, improvements and code optimization

LOG 2.3.55 WHAT IS NEW IN THIS VERSION

Just some minor fixes and improvements:

- Input Manger now has more editor options, like add new axes, and the options to set the default input, save and load works perfectly. Also, its code has been improved and simplified.
- Every custom editor button has been fixed. Now the changed values by these buttons are correctly stored and changed when the game is played.

LOG 2.3.6 WHAT IS NEW IN THIS VERSION

This is the biggest update ever for this asset and there is a lot of new features. Also, a lot of parts has been remade or amply modified. Here the full list of features for this update:

- PLAYER
  - Any number of extra jumps
  - Steps particles and footprints
  - Ragdoll when character dies or receive damage higher than x
  - IK for powers, weapons, vehicles, ….
  - Land mark on player’s feet
- CAMERA SYSTEM
  - Shake on damage
  - Headbob with states and external shakes
  - Static and dynamic headbob
  - Fixed camera positions with option to follow player (alpha)
- HEALTH SYSTEM
  - Damage screen with fade color and damage position icons
  - Damage and heal numbers in screen
  - Advanced damage detection for characters, vehicles, ….anything
  - Configure weak spots, damage multipliers and one shoot killed zones
  - Complete editor to configure easily damage receivers
- INPUT
  - Unified keyboard, touch and gamepad input
  - Enable or disable every action
  - Gamepad support with movable mouse cursor
  - Very customizable touch controls and editor configuration
  - Configurable path to save input, saves, captures, etc….
- PLAYER WEAPONS
  - Extensible and very customizable player weapon system
  - Fire in First and third person
  - Advanced use of IK: No animations needed
  - Full weapon HUD

    - Change between weapons in any camera view
    - Separated camera for weapons in first person
    - Draw, walk, aim and keep weapon actions
    - Insane amount of configurations
    - Procedural weapon sway and motion
    - Realistic physics projectile shells
    - Insane amount of options to shoot: spread, projectiles per shoot, clip size, bullet speed, bullet force amount, explosion radius,
    - Camera shake for every fired projectile
    - Smooth transition in weapon change, camera fov aiming, …
    - Procedural weapon recoil
    - 5 different weapons: pistol, regular shotgun, assault rifle, double shotgun and missile launcher
    - Pick and drop weapons ingame
    - Decals manager
    - Weapon and ammo pickups
- INVENTORY
    - Powerful inventory system
    - Pick, drop, use, lock or read info for every inventory object
    - Use objects by menu, interaction button or trigger
    - Animation in objects used
    - Complete custom editor system to configure every object added
    - Any number of slots and amount of objets per slot
    - Smart inventory management
    - Lock, rotate and zoom in every 3d model object in game
    - Easy capture manager: Use actual 3d model images
    - Increase inventory size in game with bags
- VEHICLES
    - Hovercraft, aircraft, hoverboard, sphere mode and weapon turret.
    - Camera states, set any number of camera positions
    - Hoverboard waypoint system
    - Skids manager for vehicle wheels
    - Advanced input setting for vehicles
    - Shake states on damage, accelerating, shooting, ….
- MAP SYSTEM
    - Procedural map builder, any shape is possible
    - Configure different floors and rooms
    - Map menu: change between floors, zoom, icons info, …
    - Place and remove map marks ingame
    - Configure easily any type of map icon: enemies, doors, elevators, vehicles,….
    - Map beacons and quick travel stations
    - Set map zones as hidden to be revealed when player reaches them
    - Unlock map parts with map pickups
    - Configure map zone color and map room texts
    - Map glossary according to map icons configured
    - Compass with 8 directions
- PICKUPS
    - Full pickup system for ammo, inventory, weapons, …

    - Powerful pickup manager: configure once, use anywhere
    - Breakable crates
    - Pickup drop system for crates, dead enemies, explosive barrels, ….
    - Pickup chests with configurable object spawn position
    - Random amount of pickups in drop system and chests
- INTERACTION ELEMENT IN SCENE
    - Buttons
    - Procedural doors
    - Procedural elevator
    - Hackable password panels
    - Ziplines
    - Jump platforms on trigger and on button pressed
    - Pressure plates, activate them with weight
    - Wayoint moving platforms
    - Heal and damage triggers
    - Falling platforms
    - Teleportation platforms
    - Player waypoint system, reach every point in the path!
    - Vending machines, instantiate anything
- GRAVITY
    - Change gravity for any object
    -
- PLAYERS MODE
    - Normal, jetpack, sphere controller and fly
- AI
    - Nav mesh AI system
    - AI can use weapons
    - Advanced waypoint patrols
    - Find friendly AI and give orders: wait, follow, attack and hide
    - Ragdoll when dead or damage received higher than x
    - Footstep system with particles and footprints
- SAVE SYSTEM
    - Configure any number of slots
    - Save with time played, camera capture, date and other game info
- OTHERS
    - Event trigger system, call any amount of functions when player picks a weapon, lock a room when he enters, ….everything you need
    - Physics system to grab, rotate, zoom, drop and throw objects.
    - Create any system with health and destructible, like a rotating laser trap!
- POWERFUL CUSTOM EDITOR
    - Most systems have an advanced editor allowing a perfect workflow.
    - Editable list: change order, remove, add, clear,….
    - Tag and layer manager: add and remove as you need, import project is not a problem anymore

LOG 2.3.6a-b WHAT IS NEW IN THIS VERSION

Some improvements and fixes and a better Character Creator with a custom Editor Window.

IMPORTANT: THE CLOSE COMBAT ANIMATIONS (KICK AND PUNCHES) ARE NOT INCLUDED IN THIS ASSET, BUT THE SYSTEM THAT USE THEM. YOU CAN FIND THOSE COMBAT ANIMATIONS AND MORE IN THE FREE ASSET TAICHI CHARACTER PACK, WHICH WAS USED TO TEST THIS SYSTEM. TO USE IT, JUST SET THE ANIMATIONS IN THE COMBAT LAYER OF THE ANIMATOR.

FROM HERE, THE DOCUMENTATION BELONGS TO THE PREVIOUS VERSION OF 2.3.6, BUT EVERY SYSTEM WILL BE EXPLAINED AND ADDED HERE VERY SOON (ALONG WITH VIDEO TUTORIALS). SOME PARTS ARE THE SAME AND ANOTHER HAVE CHANGED A LOT OR THEY ARE NEW. IN THE MEAN TIME, CHECK THE DEMO SCENE AND THE PREFABS, MOST OF THEM ARE MADE TO BE REALLY EASY TO USE AND CONFIGURE.

SORRY FOR THE PROBLEMS, THE LAST UPDATE 2.3.6 IS HUGE AND IT WAS RELEASED AS BETA MOSTLY DUE TO THIS.

I WILL HAVE FREE TIME IN A FEW WEEKS, AND THEN THIS FILE WILL BE COMPLETE.

CONTROLS

By default, these are the input controls:

	PLAYER	**VEHICLES**
WASD	Move character	Move to left, right, back and forward
Mouse	Move Camera	Move Camera
Space	Jump	Jump

R	One = Active gravity power Two = Change gravity in camera direction Three = stops in the air again	One = Change gravity in camera direction
F	Deactivate gravity power	Deactivate gravity control
Shift	In the air: Accelerate air movement	In the air: Accelerate air movement

	In the ground:
	One = Run
	Press again = disable run
	Hold = Run in any Surface
	Up = disable run
		In the ground:
		Hold = Activate boost
		Release= Disable boost

E	(also in aim mode)
	One=grab objects
	Two=drop objects
	Hold and release=launch objects	

Mouse Wheel	Change powers	Change between weapons
Mouse Left button	Use power/Punch/Change object gravity	
Mouse Right button	Kick/Launch enemy bullets	Fire weapons
G	Activate/Deactivate aim mode	Break
Z	Activate/Deactivate time bullet	Activate/Deactivate time bullet
C	Change between touch and keyboard controls	
V	Change between first and third camera	Change between first and third camera
Q	Activate/Deactivate shield	
T	Activate devices	Get off from vehicle
X	Crouch	
Left Ctrl	Activate/deactivate move away of the camera	Activate/deactivate move away of the camera
P	Select next power	Select next weapon
O	Select previous power	Select previous weapon
Left Alt	Enable/disable edit powers manager	

\	Hold=Enable select powers manager Release=Disable select powers manager	
Numbers keys	Select powers	Select powers
Esc	Activate/Deactivate menu	Activate/Deactivate menu

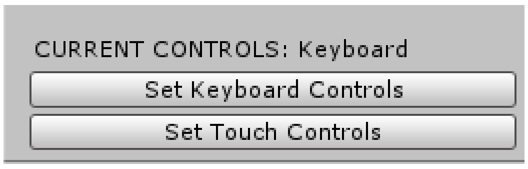
TOUCH MOBILE CONTROLS

The touch controls can be used in three ways in the asset:

- To try it in the editor while you are developing a game or making a level.
- In a PC build, for the same purpose.
- In mobile devices, android and ios.

The input is made in a way that the touch controls can be tested in PC using the mouse as a finger.

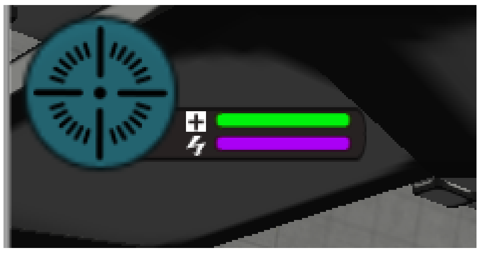
Now, if you want to activate the keyboard or touch controls, go to Character inspector, open the Input Manager inspector and press Set Keyboard controls or Set Touch Controls. Also, a label shows the current selected option. When you press play, the controls will be activated according to that option.


CURRENT CONTROLS: Keyboard

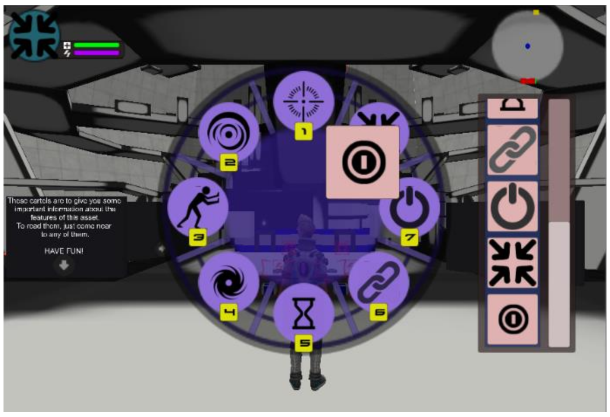
As a new feature, when you compile the asset for pc or mobile, the code checks the current platform where the game is being played. Even if the platform is PC and the touch controls are enabled and vice versa.

PLAYER

When touch controls are enabled, you can swipe your finger in the upper left corner to change between powers.

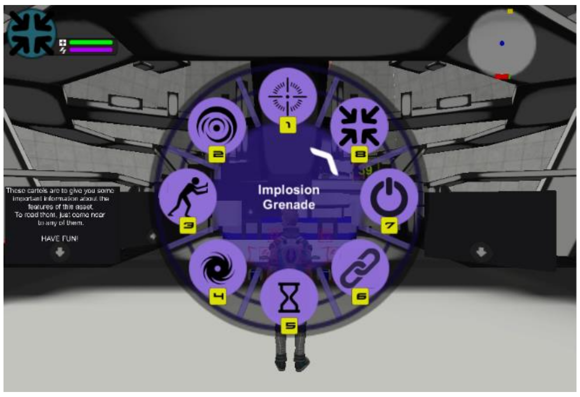


Also, if you make two quick taps in the upper left corner of the screen the edit powers manager is enabled, so you can change the current enabled powers. Make to quick taps again to disable the menu.

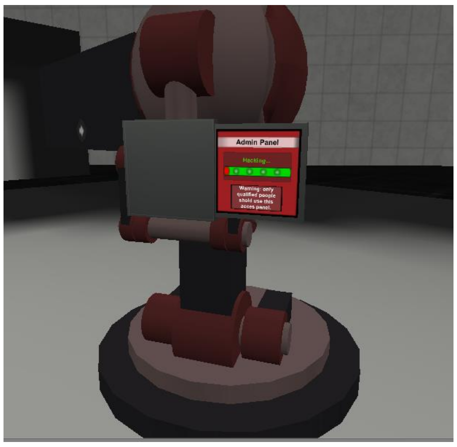

These cartels are to give you some important information about the features of this asset.
To read them, just come near to any of them.

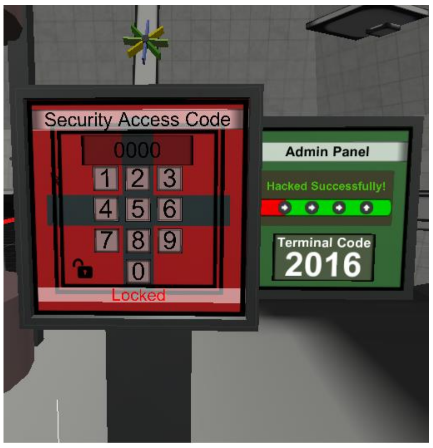
HAVE FUN!

If you hold your finger in the center of the screen, you can enable the select powers menu, move your finger to the power that you need and release the tap.


Implosion Grenade
These cartels are to give you some important information about the features of this asset. To read them, just come near to any of them.
HAVE FUN!

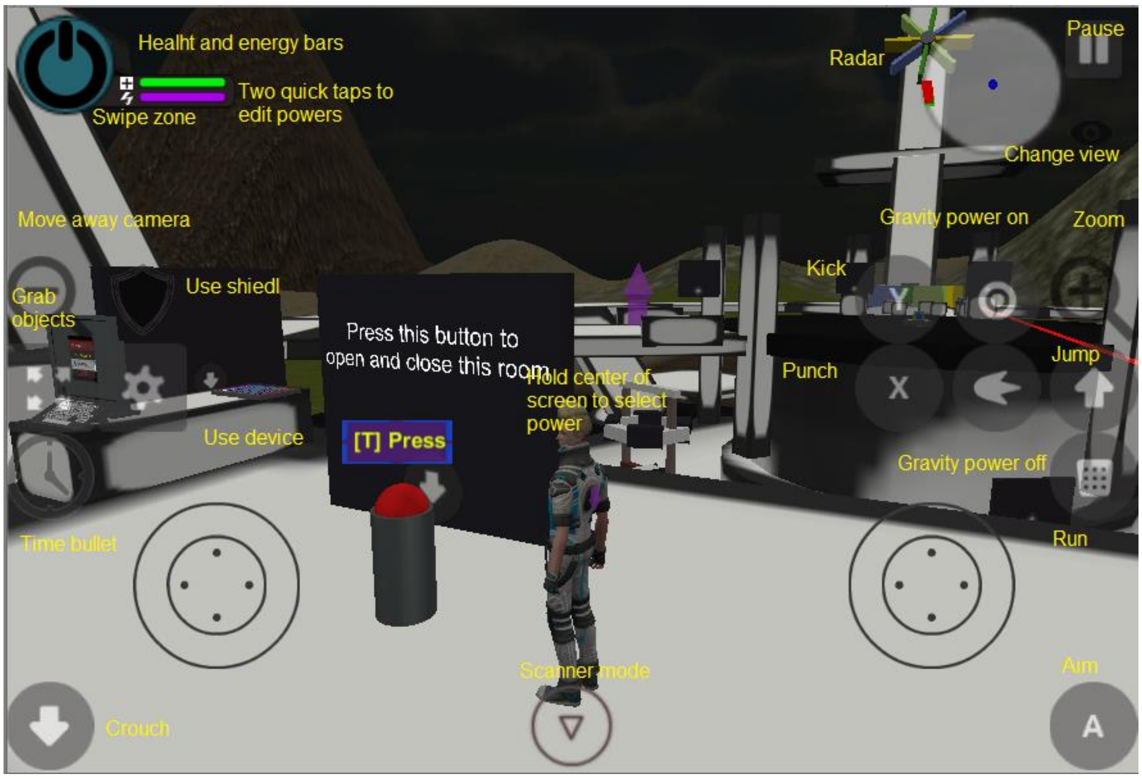
Also, when you are hacking a device or an enemy, you have to swipe in the screen, in the direction of every arrow correctly, to hack successfully that device.



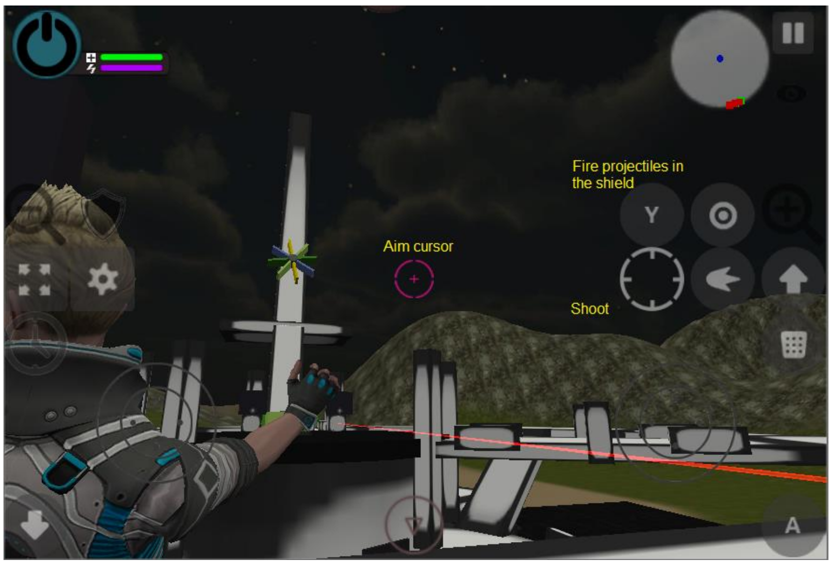


In the Touch options menu, you can set the position of every touch button and save them.

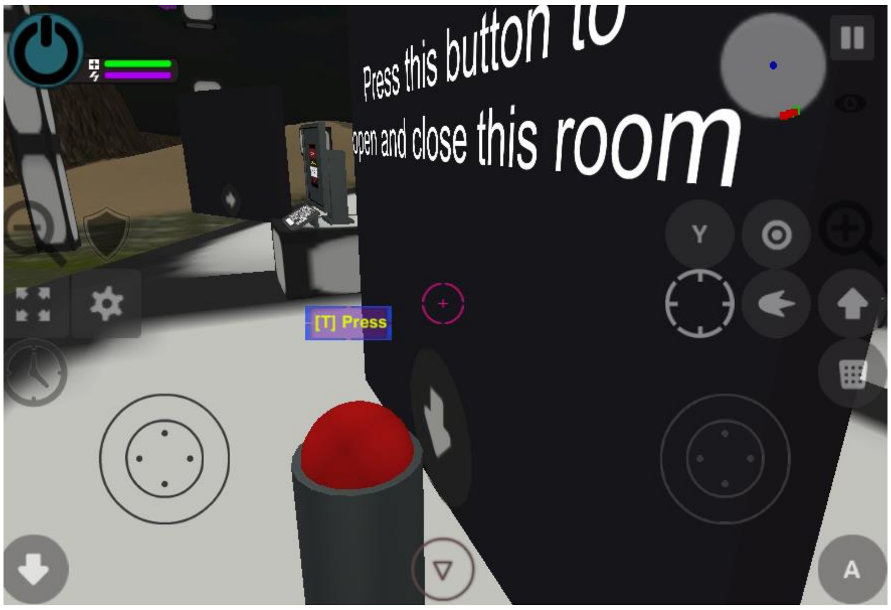
This is the use of every button in touch mode:



In aim mode, the close combat buttons are changed for the shoot button:

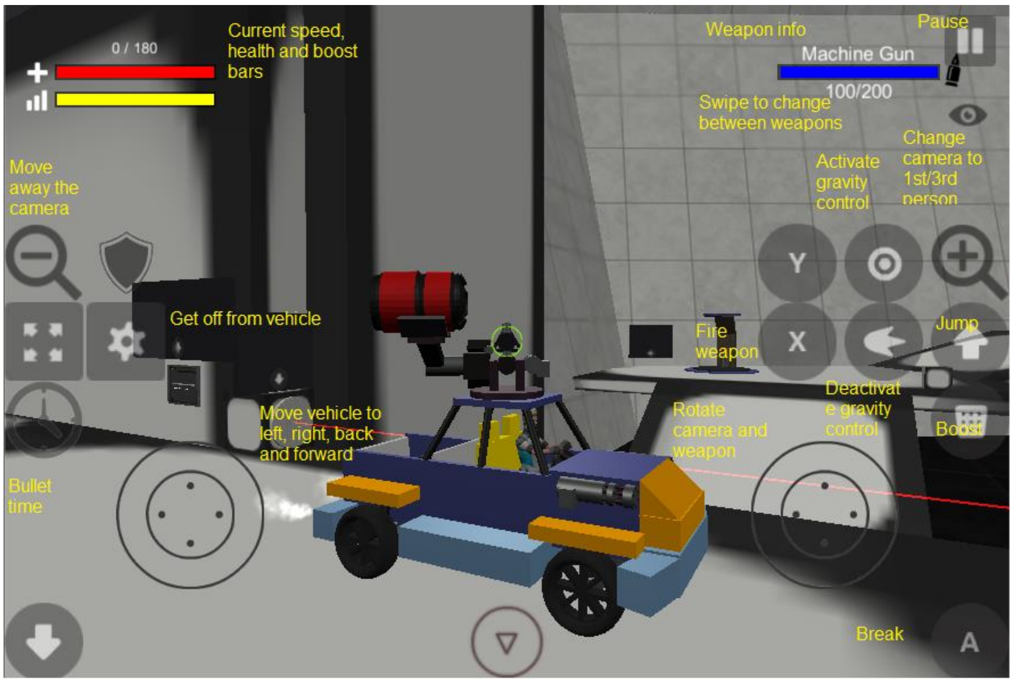


Just like the in the first person view:

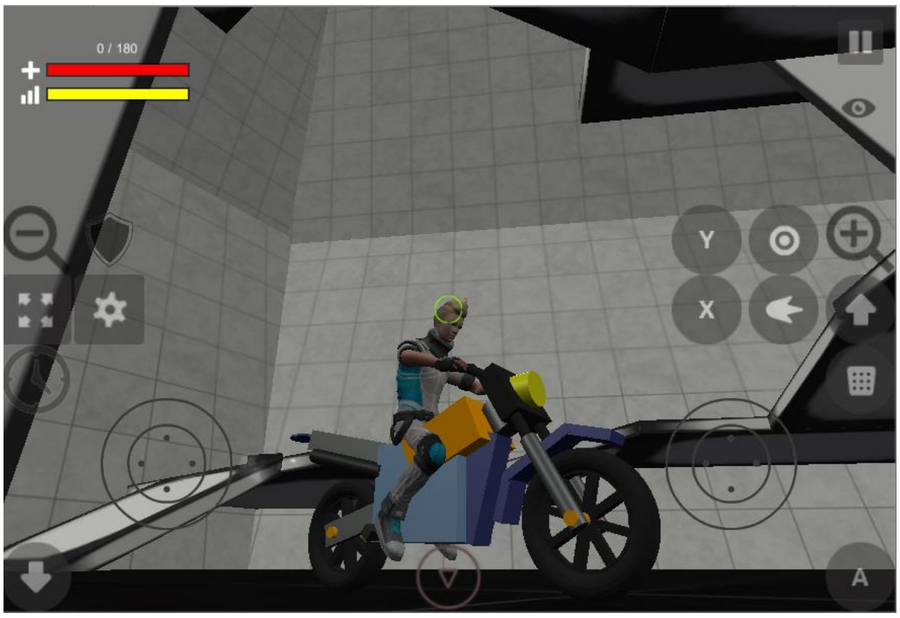


VEHICLES

When you are driving a vehicle, the touch buttons are the same, with a similar use to the player controls, but some of them don't have any use. Also, the HUD for vehicles is different from the one used for the player. In future updates, the touch controls will have different aspect for player and vehicle control.


0 / 180
Current speed, health and boost bars
Weapon info
Pause
Machine Gun
100/200
Swipe to change between weapons
Move away the camera
Activate gravity control
Change camera to 1st/3rd person
Y
Get off from vehicle
Fire weapon
X
Jump
Move vehicle to left, right, back and forward
Rotate camera and weapon
Deactivate gravity control
Boost
Bullet time
Break
A

When the vehicle hasn't a weapon system, the upper right info is disabled, and the controls are the same without the weapons.



JOYSTICK VS TOUCHPAD

The joysticks has different options to configure. Left and right controls can be used as joysticks or touchpads. The difference is that with joysticks, the movement is applied constantly when the finger moves in any direction and with touchpad, you have to keep moving your finger to rotating the camera or move the player.

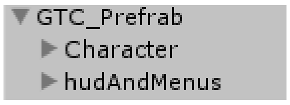
This option can be changed in the editor and ingame in the touch options menu for both joysticks.

MAIN SCRIPTS

In the version 2.3.5, there are different scripts for every object. Most of them are new, and other have been improved and simplified. The more important are:

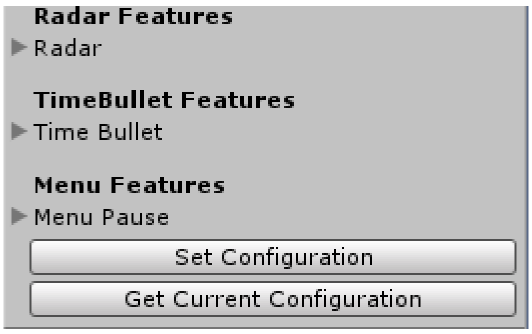
GTC_PREFAB

Hierarchy order, inside the GTC_Prefab there are Character with all the parts of the player, and hudAndMenus, which is the canvas component that contains all the UI elements used in the HUD.



▼ GTC_Prefrab
  ▶ Character
  ▶ hudAndMenus

FEATURES MANAGER, it allows to enable and disable every feature in the asset, setting every Boolean to true or false, so every script will check these values, to use the functions of the feature or not when the scene is played.



Radar Features
▶ Radar

TimeBullet Features
▶ Time Bullet

Menu Features
▶ Menu Pause

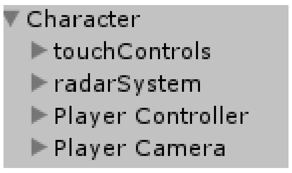
Set Configuration

Get Current Configuration

- **Set configuration**, once you have enable or disable any feature of the inspector, press this button to save that configuration in every component of the character.
- **Get current configuration**, if you have enabled or disabled features in the single scripts, you can use this button to get the current asset configuration, to see what features are actually enabled and disable.

CHARACTER

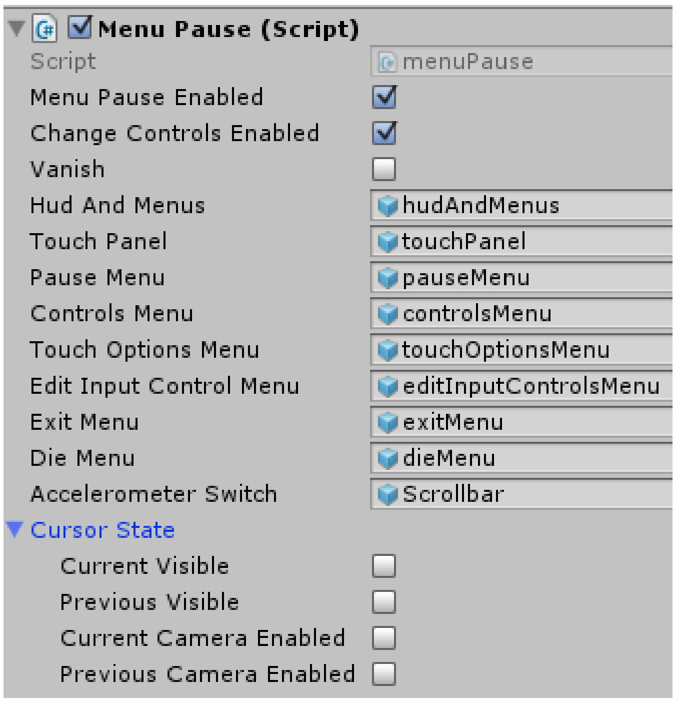
Hierarchy order, it contains the touch joysticks, the radar system, the player controller and the player camera.



▼ Character
  ▶ touchControls
  ▶ radarSystem
  ▶ Player Controller
  ▶ Player Camera

TIME BULLET, it slows the time, in the scale of its parameter by pressing Z.

MENU PAUSE, a UI menu with some options. If freezes the time scale, disable some scripts to avoid any problem. To show or hide this menu, press Esc. You can set if the scene fades or not when you execute the scene. You can change the controls type pressing C. The script has a mouse manager, to lock and unlock, and to make visible or invisible the mouse, according to if the menu is active, the touch controls are enabled, a device is being used, etc….



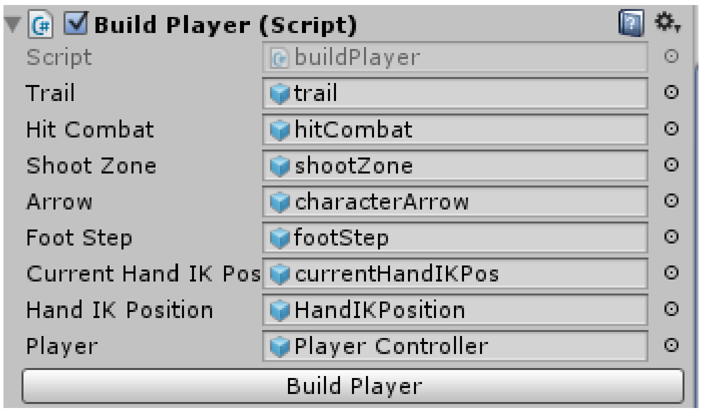
Menu Pause (Script) ☑

Script	menuPause
Menu Pause Enabled	☑
Change Controls Enabled	☑
Vanish	☐
Hud And Menus	hudAndMenus
Touch Panel	touchPanel
Pause Menu	pauseMenu
Controls Menu	controlsMenu
Touch Options Menu	touchOptionsMenu
Edit Input Control Menu	editInputControlsMenu
Exit Menu	exitMenu
Die Menu	dieMenu
Accelerometer Switch	Scrollbar
Cursor State	
Current Visible	☐
Previous Visible	☐
Current Camera Enabled	☐
Previous Camera Enabled	☐

- **Vanish**, to make the UI alpha color changes from 1 to 0, when the game starts.
- **Hud and menus, touch panel, pause menu, controls menu, touch options menu, edit input control menu, exit menu and die menu** are the elements inside hudAndMenus, to enable and disable all these elements.
- **Cursor state**, used to see the current and previous state of the camera and the cursor.

BUILD PLAYER, a script to set your own character inside the player, simplifying the assignment of the objects inside the character. It searches every bone inside the skinned mesh renderer inside the model, and set the position of every gameObject needed like the footsteps triggers, the shoot zone to fire projectiles, the triggers in hands and feet for the combat system, the trail renderers used when the player runs and the arrow in the back. Also it sets all the configuration in others scripts that use them. With the new aim mode introduced in 2.3.5, it has two more objects to place, the current Hand IK Pos, to set the current position of the current hand used for the aim mode and the Hand IK Position to set the place where every hand will be placed and rotated in the aim mode.

This script allows to set your own character model just by dropping the model inside the gravity center inside the player controller.

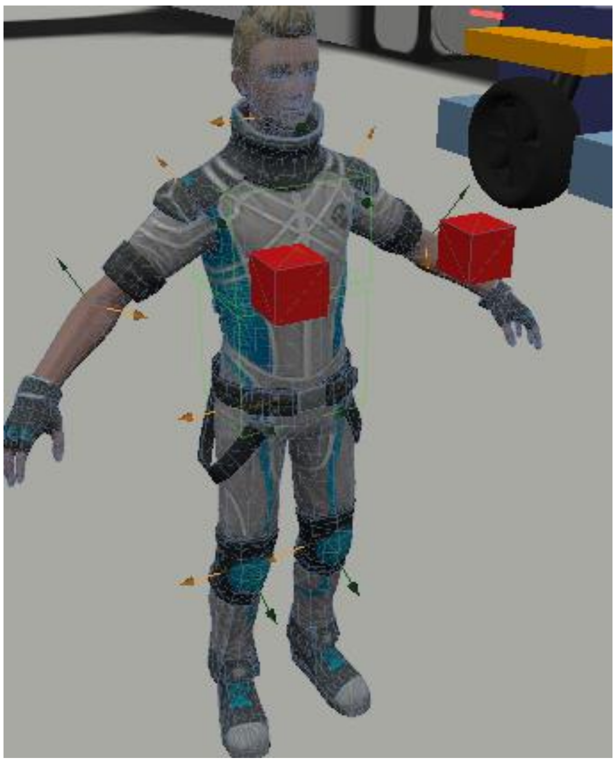


Build Player (Script)

Script	buildPlayer
Trail	trail
Hit Combat	hitCombat
Shoot Zone	shootZone
Arrow	characterArrow
Foot Step	footStep
Current Hand IK Pos	currentHandIKPos
Hand IK Position	HandIKPosition
Player	Player Controller

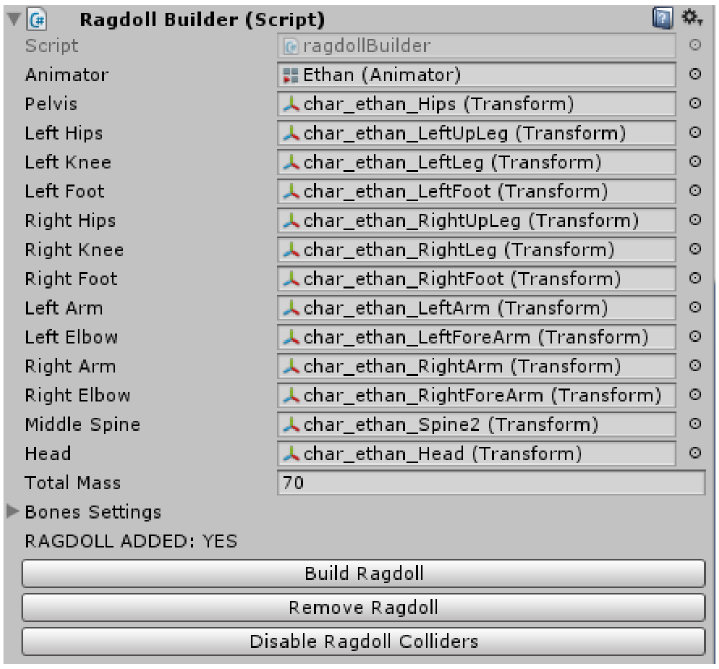
Once that only your 3d model is inside gravityCenter, press the button Build Player. This script also sends the animator component to the ragdollBuilder to set the player's ragdoll.

Once the model is build, you will get this:



The red boxes are used to set the position where every hand of the player will be placed in aim mode, using IK, so just move them where you need.

RAGDOLL BUIDLER, once that your model is placed and the build player button has been pressed, you can add a procedural ragdoll to the character with this inspector, instead of using the ragdoll wizard or unity, but both options can be used.


Ragdoll Builder (Script)

Script	ragdollBuilder
Animator	Ethan (Animator)
Pelvis	char_ethan_Hips (Transform)
Left Hips	char_ethan_LeftUpLeg (Transform)
Left Knee	char_ethan_LeftLeg (Transform)
Left Foot	char_ethan_LeftFoot (Transform)
Right Hips	char_ethan_RightUpLeg (Transform)
Right Knee	char_ethan_RightLeg (Transform)
Right Foot	char_ethan_RightFoot (Transform)
Left Arm	char_ethan_LeftArm (Transform)
Left Elbow	char_ethan_LeftForeArm (Transform)
Right Arm	char_ethan_RightArm (Transform)
Right Elbow	char_ethan_RightForeArm (Transform)
Middle Spine	char_ethan_Spine2 (Transform)
Head	char_ethan_Head (Transform)
Total Mass	70

Bones Settings

RAGDOLL ADDED: YES

First, set the **total mass** of your character and press build ragdoll. Instantly you can see the colliders with joints and rigidbodies attached to the skeleton of the model. The label **RAGDOLL ADDED** will show if the character has a ragdoll in it or not.

After that your model is ready to use the ragdoll when player dies, if you uses the ragdoll option instead a death animation.

Another option is to press the **disable ragdoll colliders**, just to disable their components in the scene. This is optional too, due to when the scene is played, the ragdoll is disabled at the beginning, avoiding any problem with the player's collider.

Finally the last option is **remove the ragdoll**, deleting the collider, rigidbody and joint components of the character. This can be used to remove the ragdoll if any part is not configured correctly or you will only use death animations.

In any collider has to be rescaled, select the bone in the inspector and edit the collider directly.

The **bones settings** allow to configure the scale or every limb collider and the percentage of mass of them. If your model is stronger or skinnier, configure these values, to set a correct proportion. The total mass can be configured without needing of configure these values.

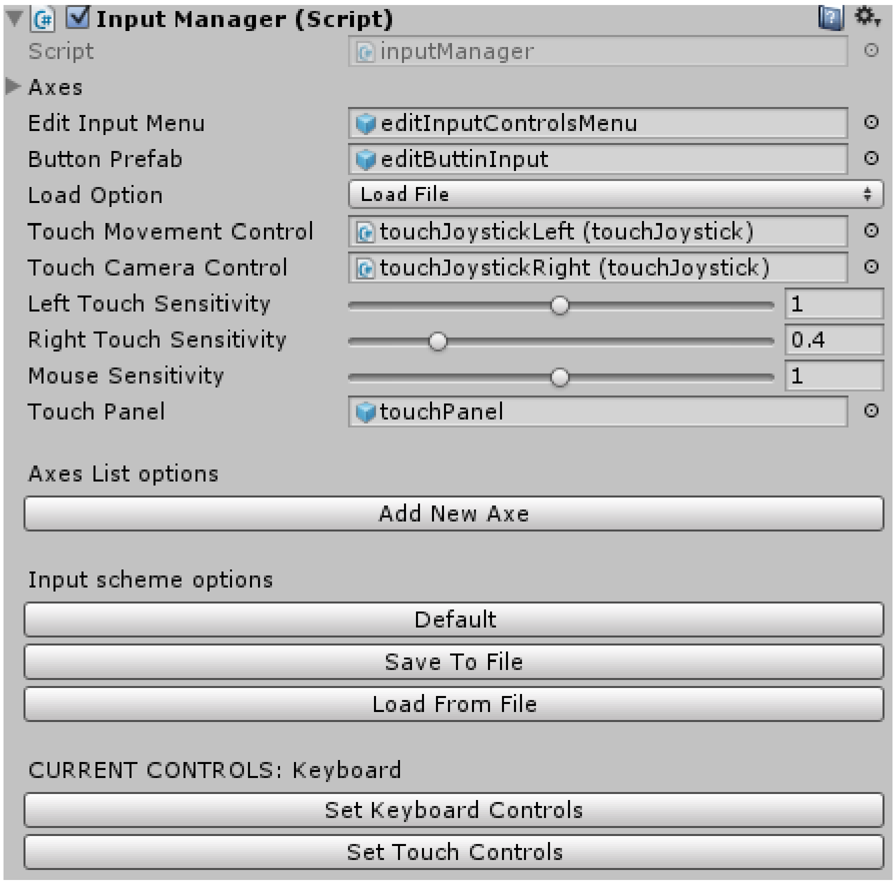
INPUT MANAGER, it allows to define a list of input keys, save this configuration, load it, and set the default controls, defined inside the script. It uses the default configuration of axes in Unity. Also, it has the main key detection system, to check if an action key defined in the axes is pressed, held and released, using the get button functions placed in the other scripts. At the same time, it checks the touch controls, asking if the action key has been pressed, held and released.

If you want to add an action to a new script, add a new Axe in the Axes list, declare a inputManager variable type, get the main inputManager in the Character gameObject and add the above functions to the script, setting the type of press button and the action that you want to check. In this example you can see it better:

```
using UnityEngine;
using System.Collections;
inputManager input;
GameObject character;
public class yourScript : MonoBehaviour {
	void Start () {
		character = GameObject.Find ("Character");
		input = character.GetComponent<inputManager> ();
	}
	void Update(){
		if (input.getButton ("newAction", inputManager.buttonType.getKeyDown) ||
		input.getTouchButton ("newAction ",
		inputManager.buttonType.getKeyDown)){
			//KeyDown, key pressed, call the need function
		}
		if (input.getButton ("newAction", inputManager.buttonType.getKey) ||
		input.getTouchButton ("newAction ", inputManager.buttonType.getKey){
			//Key, key held call the need function
		}

		if (input.getButton ("newAction", inputManager.buttonType.getKeyUp) ||
		input.getTouchButton ("newAction ", inputManager.buttonType.getKeyUp)){
			//KeyUp, key released, call the need function
		}
	}
}
```

The code contains both functions to check the keyboard and the touch screen, but the input manager script only checks the current control, not both and the same time, according to if the player is using a PC or a smartphone/tablet.


Input Manager (Script)

Script	inputManager
Axes	
Edit Input Menu	editInputControlsMenu
Button Prefab	editButtinInput
Load Option	Load File
Touch Movement Control	touchJoystickLeft (touchJoystick)
Touch Camera Control	touchJoystickRight (touchJoystick)
Left Touch Sensitivity	1
Right Touch Sensitivity	0.4
Mouse Sensitivity	1
Touch Panel	touchPanel

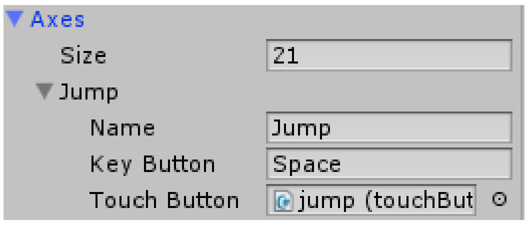
Axes List options

Input scheme options

CURRENT CONTROLS: Keyboard

- **Load option**, is used to select if you want to load the axes from the saved file, or from the values in the axes list in the inspector.
- **Touch movement and camera control**, here are located the touch joysticks. Any other script can ask to this component the current values from the joysticks input and the horizontal and vertical axis. Vehicles use this axis values to be driven for example.
- **Left and right touch sensitivity**, the sensitivity used for every joystick in the touch control.
- **Mouse sensitivity**, the sensitivity used in the mouse
- **Add New Axe**, this will add a new axe button to the list in the inspector.
- **Default**, set the default input values in the axes list. It is the initial configuration of the asset. It deletes the current axes list and create a new one, assigning every touch button too.
- **Save to file**, save the current axes list to a file, so that it is the current input.
- **Load from file**, load from the saved file the axes list.
- **CURRENT CONTROLS**, is a label to show if the current controls are keyboard or touch.
- **Set keyboard controls**, it sets the current controls of the game to the keyboard input.
- **Set touch controls**, it sets the current controls of the game to touch input.

These two last options are used for test modes in the editor, when the game is built, the code checks and set the current platform controls, keyboard or touch.

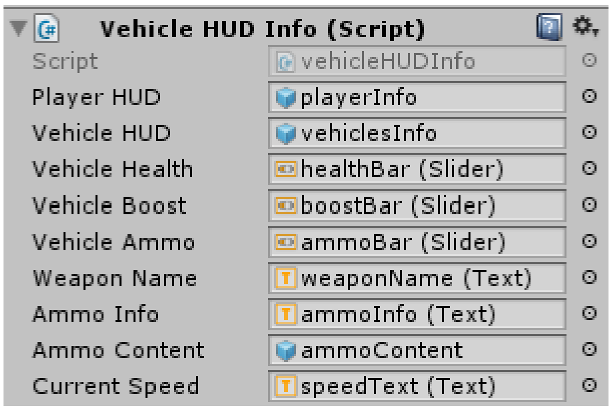


▼ Axes	
Size	21
▼ Jump	
Name	Jump
Key Button	Space
Touch Button	jump (touchBut

- **Axes**, in this list, you define an action, set a key to make the action. The touch button can be add if you use a touch control, but it can be as an empty field. To define a new control, increase the size of the list in 1. IMPORTANT, once that you have added a new axe, press the option save to file, to avoid issues.

VEHICLE HUD INFO, this script allows to store every HUD component used for vehicles and the player's HUD as well, because all the vehicles use only one HUD, the same for all of them, instead of using one different HUD for every vehicle.

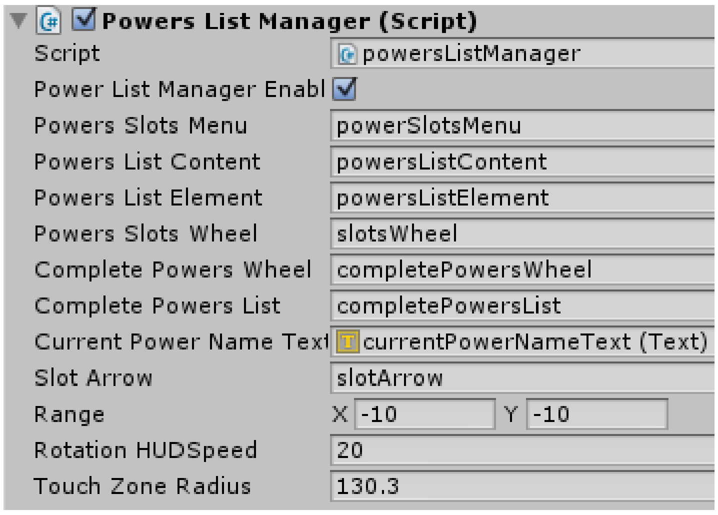
So when the player gets in to the vehicle, its HUD manager component gets the info stored in this script, so the info in the vehicle HUD is updated with the info of the vehicle driven by the player in that moment. Also, the player's HUD is disable until the player gets off from the vehicle.



Vehicle HUD Info (Script)	
Script	vehicleHUDInfo
Player HUD	playerInfo
Vehicle HUD	vehiclesInfo
Vehicle Health	healthBar (Slider)
Vehicle Boost	boostBar (Slider)
Vehicle Ammo	ammoBar (Slider)
Weapon Name	weaponName (Text)
Ammo Info	ammoInfo (Text)
Ammo Content	ammoContent
Current Speed	speedText (Text)

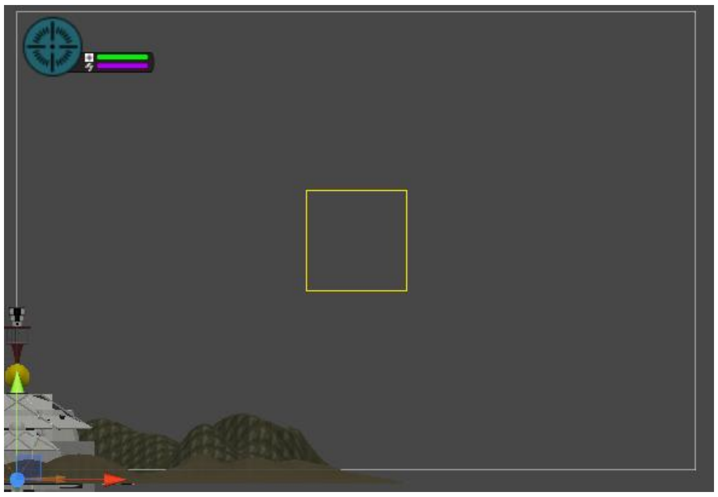
PLAYER STATES MANAGER, used to get the current state of the player before he uses a vehicle and disable it in case he is doing it, like the aim mode, drop any object that he could carry, disable the scanner, and any other action that could create problems.

POWER LIST MANAGER, set the list of powers defined in otherPowers in the powers wheel menu and the list if the right side. It allows to edit the order and the current power list that the player uses and also selecting them moving the mouse.



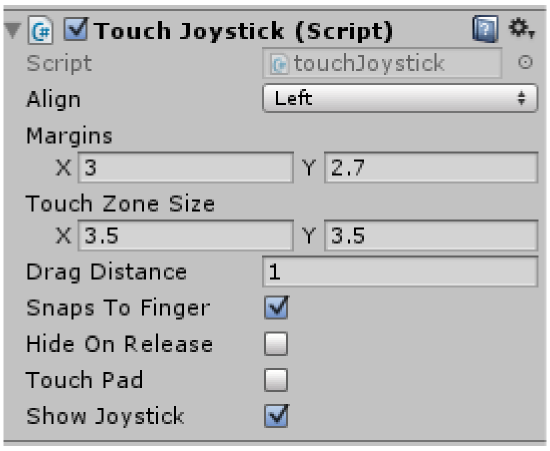
Powers List Manager (Script)	
Script	powersListManager
Power List Manager Enabl	✓
Powers Slots Menu	powerSlotsMenu
Powers List Content	powersListContent
Powers List Element	powersListElement
Powers Slots Wheel	slotsWheel
Complete Powers Wheel	completePowersWheel
Complete Powers List	completePowersList
Current Power Name Text	currentPowerNameText (Text)
Slot Arrow	slotArrow
Range	X -10 Y -10
Rotation HUDSpeed	20
Touch Zone Radius	130.3

- **Range**, it set the amount of rotation that the powers wheel makes following the mouse position when the player is selecting a power.
- **Rotation HUDSpeed**, how fast the powers wheel rotates
- **Touch Zone Radius**, the size of the rect zone in the center of the screen to hold the finger to enable the select power menu. If you select the Character gameObject, and look the scene window, you will see the size of the rect.



TOUCH CONTROLS

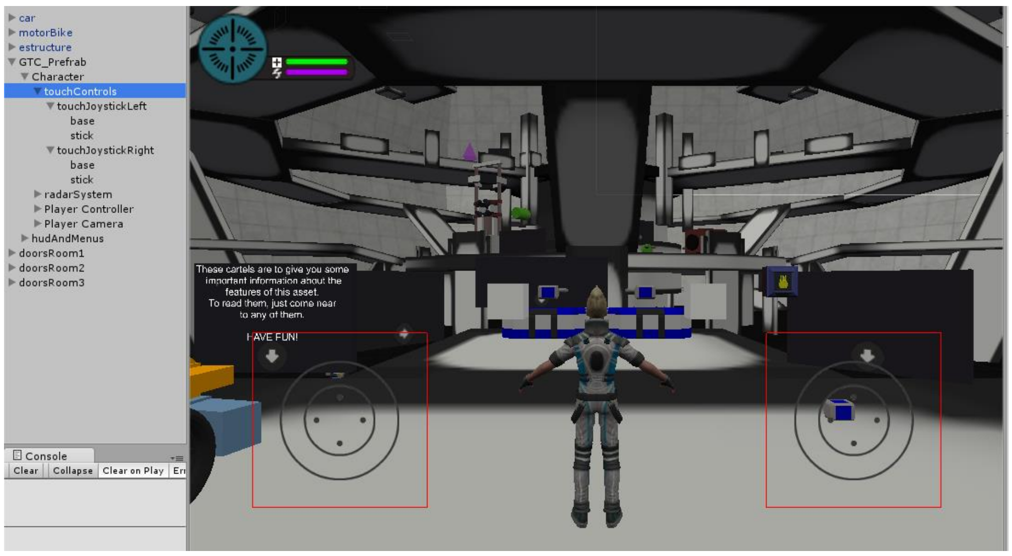
TOUCH JOYSTICK, this is the script to control the move of the player in touch devices. It has some configurations to set the margins, its alignment, touch size, etc… Every joystick has to be inside a camera with orthographic projection, to be visible in the main camera, and inside the joystick there are the stick and the base, to show the control position when the player are using it. This script allows move the joysticks in pc, using the mouse input like a finger tap.


Touch Joystick (Script)

Script: touchJoystick
Align: Left
Margins: X 3, Y 2.7
Touch Zone Size: X 3.5, Y 3.5
Drag Distance: 1
Snaps To Finger: ✓
Hide On Release: ☐
Touch Pad: ☐
Show Joystick: ✓

- **Align**, select the side of the screen to display the joystick, left or right.
- **Margins**, the horizontal and vertical spaces between the joystick and the lower corner of the screen.
- **Touch zone size**, the touch rect size of the joystick, to check if the player touches inside the joystick zone, to be able to move it.
- **Drag distance**, the distance to drag the joystick once is grabbed by the finger.
- **Snaps to finger**, if true, when the player touches in the joystick touch zone, the position of the joystick is changed below the player’s finger.
- **Hide on release**, if true, if the joystick is not being used, it is not visible, else the joystick is visible while the player is using it.
- **Touchpad**, if true, the joystick control is used as a touchpad, so the axis value input is only apply once for the object that use it, instead of applying the axis value constantly like the joystick does.
- **Show joystick**, if true, the joystick icons are showed in the screen, else it is not showed.

For a better configuration of the joysticks, you can enable it in the hierarchy and configure the margins to set a bigger or smaller touch zone detection, so if the finger press inside of this touch zone, the joystick gets the finger movement and translate it to input values. Then select the touch controls parent in the hierarchy and enable the Gizmos option in the game window to see the touch zone size of both controls, like this picture with the default configuration.

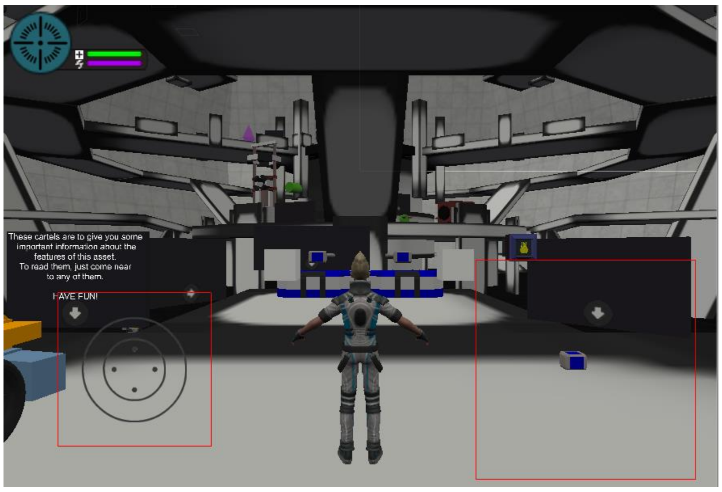


These cartels are to give you some important information about the features of this asset. To read them, just come near to any of them.

HAVE FUN!

The touch zone size in red shows where the player can touchs to move the joystick, so if the control is not used as a touchpad, the touch zone size can be small.

For example if you set the left stick to visible not used a touchpad and the right stick invisible and as a touchpad, you can increase the touch zone size for the right one, so the player can has a bigger touch zone to press and move the finger, like in this picture:



These cartels are to give you some important information about the features of this asset. To read them, just come near to any of them.

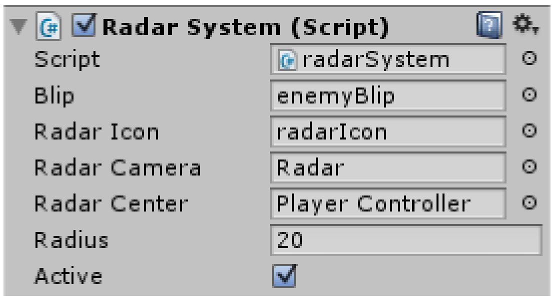
HAVE FUN!

RADAR SYSTEM

RADAR SYSTEM, it has a list of gameObjects and for every character a cube is instantiated as its icon, being only visible in a second camera, to represent every type of object in the radar: yellow-objective, red-enemy and green-ally. When a new object of that type is created, it is stored in the list of the radar, and when it is destroyed, the object is removed.

The distance from the player to every object is calculated, if the object is close enough, its icon is inside the radar, else the object is outside the radar, but visible in its border.

The layer use for this objects is radar. In the main camera, this layer is not visible.



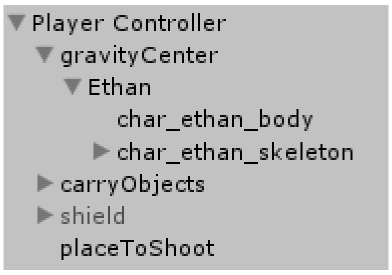
Radar System (Script)

Script	radarSystem
Blip	enemyBlip
Radar Icon	radarIcon
Radar Camera	Radar
Radar Center	Player Controller
Radius	20
Active	✓

- **Blip**, a cube used for every object in the radar.
- **Radar Icon**, the circle icon used for the radar.
- **Radar camera**, the camera used to see the radar objects.
- **Radar center**, the object that the radar follows as its center
- **Radius**, the radius of the radar to limit the cubes max distance.
- **Active**, if true, the radar is enabled, else the radar is disabled.

PLAYER CONTROLLER

Hierarchy order, it contains the gravity center of the player, and inside it the empty gameObject that contains the skinned mesh renderer of the player's 3d model and its skeleton. Carry objects is used to grab and carry objects when the player is not in aim mode. It contains the shield used by the player and the place to shoot used for the health component as the place where enemies fires, located in the center of the player's model.

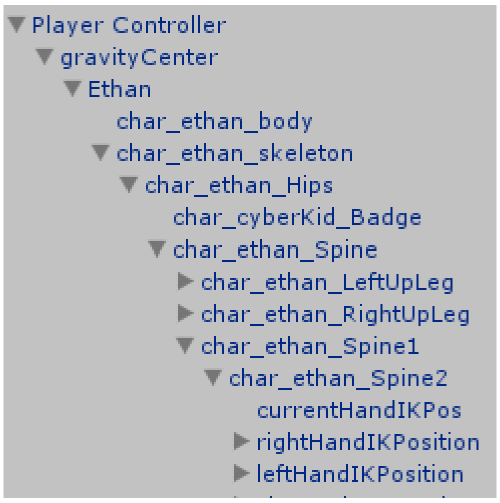


```
▼ Player Controller
  ▼ gravityCenter
    ▼ Ethan
        char_ethan_body
      ▶ char_ethan_skeleton
  ▶ carryObjects
  ▶ shield
    placeToShoot
```

Also, inside the skeleton there are more elements, like a foot steps trigger in the every foot of the player, the triggers in feet and hands for the combat, the trail renderers in different parts for the combat and the run mode and the gravity arrow, which aims to the regular ground.

With the use of IK there are there three more elements in the skeleton, the IK positions for every player's hand when the player aims and the current hand position empty transform, used set the current hand position, due to the IK positions are not moved once the game starts, they are used to set the default position of every hand.

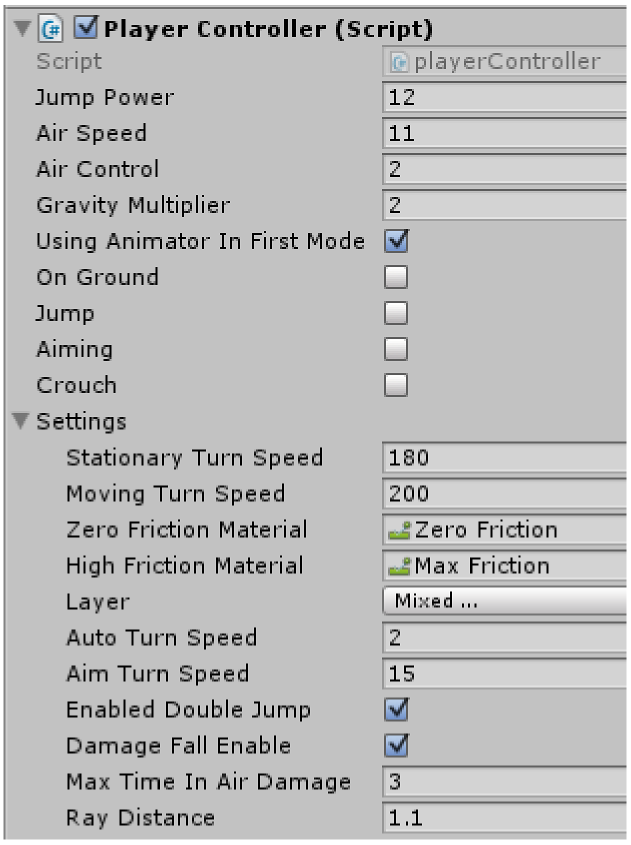
These three elements are placed inside the chest bone of the player's skeleton, showed in the last three elements of this hierarchy:



```
▼ Player Controller
  ▼ gravityCenter
    ▼ Ethan
        char_ethan_body
      ▼ char_ethan_skeleton
        ▼ char_ethan_Hips
            char_cyberKid_Badge
          ▼ char_ethan_Spine
            ▶ char_ethan_LeftUpLeg
            ▶ char_ethan_RightUpLeg
            ▼ char_ethan_Spine1
              ▼ char_ethan_Spine2
                  currentHandIKPos
                ▶ rightHandIKPosition
                ▶ leftHandIKPosition
```

PLAYER CONTROLLER, it takes the values of the keyboard and convert them to make the Mecanim works. Also convert the global movement into local, so the player is able to walk in any surface. At the same time, it checks if the player is in the ground or in the air, applying the gravity in his negative local axis Y. It allows to move the player in the air while he is jumping, having a little of control in the air. Now it has the left joystick. You can configure some values in the editor.

The player can crouch, scaling the capsule collider of the character according to his state. Also when the player aims, the character rotates towards the camera direction. Finally, when the player is hacking a device, using it or driving, the player stops his movement and it is disabled until the player ends the interaction.



Player Controller (Script)

Script	playerController
Jump Power	12
Air Speed	11
Air Control	2
Gravity Multiplier	2
Using Animator In First Mode	✓
On Ground	☐
Jump	☐
Aiming	☐
Crouch	☐
Settings	
Stationary Turn Speed	180
Moving Turn Speed	200
Zero Friction Material	Zero Friction
High Friction Material	Max Friction
Layer	Mixed ...
Auto Turn Speed	2
Aim Turn Speed	15
Enabled Double Jump	✓
Damage Fall Enable	✓
Max Time In Air Damage	3
Ray Distance	1.1

- **Jump Power**, the force applied to the player when he jumps.
- **Air speed**, the amount of movement that the player can move when the falls in the air.
- **Air control**, how much control the player has when he falls.
- **Gravity Multiplier**, the amount of force applied to his negative local Y axis.
- **Using animator in first mode**, set if you want to move the first person mode using the animator motion. Else the player moves applying velocity directly to his rigidbody.
- **On ground, jump, aiming and crouch** are variables just to check the current state of the player.
- **Stationary and moving turn speed**, the speed of rotation applied to the player when he rotates moving or quiet.
- **Zero and high friction material**, the material set to the collider when the player moves or not.
- **Layer**, use for the raycast below the player to check if he is in the ground or in the air.
- **Auto and aim turn speed**, used when the player is in aim mode and the camera rotates.
- **Enabled double jump**, to make the player can jump one or two times.
- **Damage fall enable**, the player receive damage when the falls from a high height.

- **Max time in air damage**, how much time the player is in the air before he will get damage for falling.
- **Ray distance**, is the distance of the ray used to check if the player is in ground or in the air.

CHANGE GRAVITY, is the script that allows the player choose the new gravity. Its performance is the next:

Press R to activate the power gravity. If the player is in the air, he stops in his position. Else the player is elevated above the ground. In both cases, a cursor will appear in the center of the screen. Once that the player is floating, press R again, and the player will move in the camera direction. When he finds a new surface that will be his new ground, rotating the player and the camera to it.

While the player is moving in the air, searching a new surface, you can press shift to accelerate his movement.

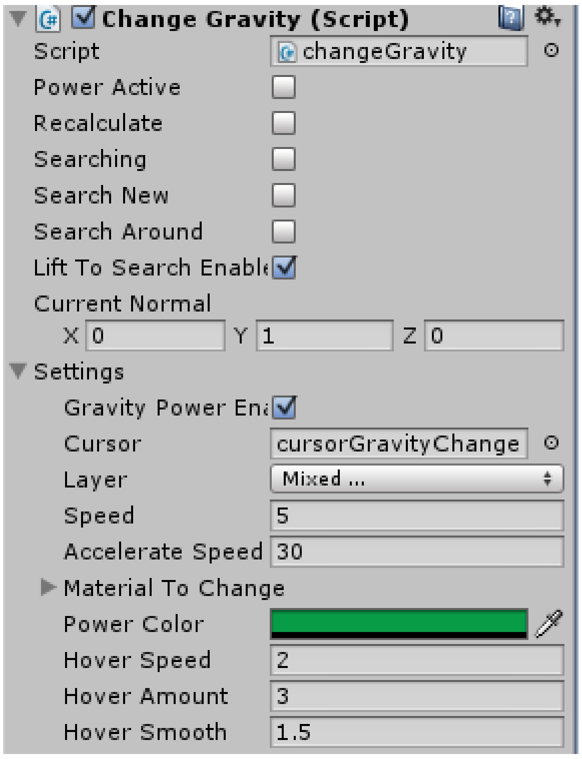
In this state the player can moves in any local horizontal direction and in any moment the gravity power can be deactivated by pressing F, whatever you are, in the air of in a new surface.

If you are in a new surface, and you fall, your gravity stills being the same, and if the player reaches certain amount of velocity, he will searches a new surface, changing his gravity to the new surface that he finds. In this state, you can also accelerate his falling speed with shift.

In the script editor, you can see in any moment the normal of the player, which is adjust every time that the player found a new surface, using the power gravity.

This script gets the root motion of the character, which is the hips. This is the object what the camera will follow, instead the playerController, when the player dies, because his body has a ragdoll which moves freely. In this process the parent of some objects are set to null, to make a smooth transition from Mecanim to ragdoll and vice versa.

In version 2.3, the gravity control has been highly improved. The player is not set to kinematic anymore, a velocity value is always applied to his rigidbody. Now the player can move while he searches a new surface. Also when the player use the gravity control, the velocity of the player is reduced smoothly, instead of setting directly to zero.



Change Gravity (Script)

Script	changeGravity
Power Active	☐
Recalculate	☐
Searching	☐
Search New	☐
Search Around	☐
Lift To Search Enabl	✓
Current Normal	X 0 Y 1 Z 0
Settings	
Gravity Power En	✓
Cursor	cursorGravityChange
Layer	Mixed ...
Speed	5
Accelerate Speed	30
Material To Change	
Power Color	🟩
Hover Speed	2
Hover Amount	3
Hover Smooth	1.5

- **Power active**, recalculate, searching, search new and search around are used just to see the current state of the player.
- **Power active**, the player has activated the gravity power, until he reachs a new surface or deactivate the gravity power, this Boolean is enabled.
- **Recalculate**, this is true when the player is circumnavigating an object, or above a rotating/moving object.
- **Searching**, true if the player is searching a surface using the gravity power.
- **Search new**, true if the player has used the gravity power, has a different normal and he falls in the air, reaching a certain amount of velocity, so the next surface that he will touch, it becomes his new ground.
- **Search around**, true when the player uses the gravity power to search a new surface. In this mode, a sphere collider in the player is enabled, so any surface that collides with that sphere collider will become the new ground. This is used, because the player can move in the air while he searches a new surface, to avoid that any surface collider could be crossed. This allows to increase the precision of the gravity power.
- **Lift to search enabled**, if true, the player is lifted when he activates the power of the gravity if he was in the ground and stops him in the air when he activates again the gravity power in the air. Else, when the gravity power button is pressed, the player directly falls in the camera direction.
- **Current normal**, just a vecto3 with the current normal of the player. At the start of the game is the regular gravity value.
- **Cursor**, the UI image used as a cursor when the players can choose a gravity direction.
- **Layer**, a layermask used for the gravity power, to detect any surface closed to the player.
- **Speed**, the amount of force applied to the player when he falls in a new gravity direction, only while he searches a new surface.
- **Accelerate speed**, the amount of extra speed added when the player press the run button, to increases his air velocity.

- **Material to change**, when the player is built, this array has the materials of the skinned mesh renderer of the player, if you set to 1, that material color will be changed while the gravity power is enabled.
- **Power color**, the color applied to the player's materials while the gravity power is enabled.
- **Hover speed**, the speed of the hover movement when the player stops his movement in the air.
- **Hover amount**, the amount of distance that the player hovers in the air.
- **Hover smooth**, how smooth is the hover movement.

OTHER POWERS, this scripts allows the player to run, when he is in the ground and moving, just press shift once to make the player run. Press again to stop of run. If you hold shift, the player also will adhere to the surface in front of him while he runs. To stop, just release shift.

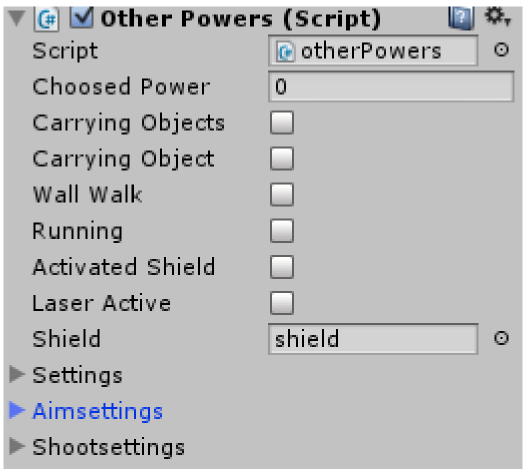
Also, this scripts allows you to grab objects in a certain distance. Press t, and the objects will be grabbed and carried at the left and right of the player, and they will move with him. If you press t again, the objects will be dropped. If you hold and release, the objects will be launched in the camera direction, with a force proportional to the duration of the holding. While you hold the key T, a UI slider will be enabled in the screen, to show how much force has been accumulated.

Other functions are:

- **Grab allies**, now the player can grab ally turrets, to move them with him and drop them where he wants. When a turret is dropped and it collides with a surface, the turret is set to kinematic, just like its previous state.
- **The power selection**, allows the player to change between powers, with the wheel mouse, the keyboard numbers, and by swiping in the left upper corner of the screen, when the touch controls are enabled. In the scene window, a rect is drawn in that position, so the player can see the swipe zone, and configure its size, due to the swipe has to start inside this rect zone.
- **The aim mode**, by pressing G, the player enable and disable the aim mode. Then, the bones of the player arms are changed to a fix rotation, as his spine, to rotate in the camera direction. Also, the camera is moved from its regular position, to the right or left shoulder of the player. To change the shoulder which the camera moves, press V.
- **The power shoot**, in the aim mode, the player can shoot by pressing the left mouse button. The direction of the bullet is the camera direction if a raycast does not find a surface, else, the rotation of the bullet is the direction from its position to the hit point. Every bullet is instantiated from the hand of the player, right or left according to the side where the player aims. Also other powers use the same hand to grab objects by pressing E.
- **Shield**, by pressing Q, the shield is activated. It rotates towards the camera direction. It protects the player from enemy projectiles, which are captured, and can be fired toward the enemies that shoot them, if they have been destroyed, by pressing the left mouse button. Else, the projectiles are fired in the camera direction. Also, when the character is touched by a laser device, the player can project a laser from his hand in aim mode, to use it as a weapon or to activate other devices.
- **Join objects**, this power allows the player shoots two objects, to add velocity to every object in the direction of the other. For example a rigidbody can moves toward a wall,

to an enemy, a rigidbody to another rigidbody, even use a rigidbody to damage an enemy. A couple of particles system are used to show the direction of every object.

- **Push objects**, using an overlap sphere, the objects in front of the player are stored in an array to add an explosion force in the camera direction, so any rigidbody will be push.
- **Grab objects**, the player can grab some objects in the normal mode, in both sides of the player. Also, the player can grab one object in the aim mode, by pressing E, so the object will be moved in front of the player. In this state, the player can drop the object, throw it and change its gravity pressing the left mouse button.
- **Change object's gravity**, this makes the objects move in line straight in the direction of the camera when the object was dropped, until it collides with any surface, when a raycast will get the normal of that surface, turning the gravity direction of the object using that normal. To set the gravity of the object to its regular state, grab the object again and change its gravity aiming to the ground. When the object will collide with the ground, its gravity will be regular again. The gravity of the ally turrets can be changed too, to set them in any surface.
- **Set damage direction**, when the player is being damaged, a panel in the screen fades to red, according to the remaining health of the player. Also an arrow icon rotates towards the enemy who is shooting. Finally, after a while without being damage, the arrow is disabled, and the red screen fades.
- **Health and power regenerative system**, it controls the health and power amount. Every power uses a different amount of power, which is subtracted from the remaining power of the player. The shield also use this power to work. When the power is not being used for a while, this will increase, until reach its max value.
- **Change global gravity**, allows to change the physics gravity values of the scene, so any regular rigidbody will fall in the new direction. Just press the shoot button and the detected normal will be the new gravity value. Shoot to the ground to set the gravity to its regular value.



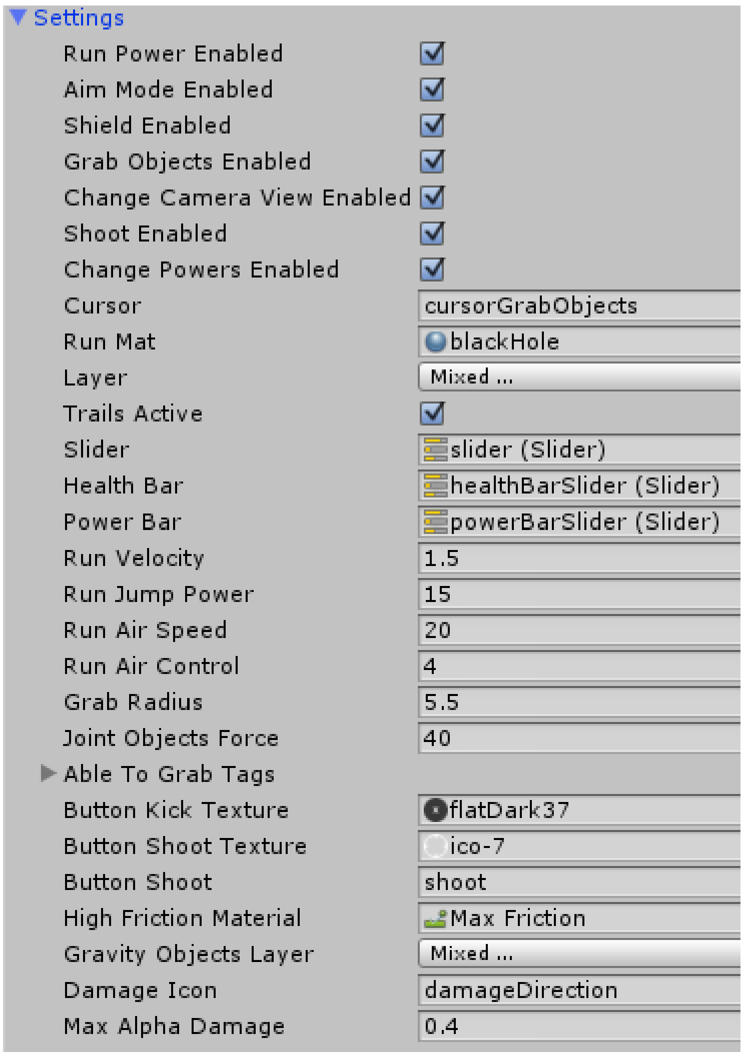
Other Powers (Script)

Script	otherPowers
Choosed Power	0
Carrying Objects	☐
Carrying Object	☐
Wall Walk	☐
Running	☐
Activated Shield	☐
Laser Active	☐
Shield	shield

▶ Settings
▶ Aimsettings
▶ Shootsettings

- **Choosed power**, set the initial power used.
- **Carrying objects**, carrying object, wall walk, running, activated shield and laser active are Booleans to see the current state of the player.
- **Carrying objects**, the player has grabbed objects in the regular mode, and he carried them around him.
- **Carriying object**, the player has grabbed an object in aim mode.
- **Wall walk**, the player is running adhering to every close surface.

- **Running**, the player is running.
- **Activated shield**, the player's shield is enabled.
- **Shield**, the gameObject that the player uses as shield

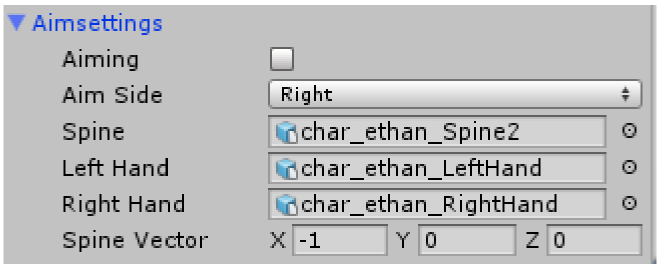


▼ Settings

Run Power Enabled	✓
Aim Mode Enabled	✓
Shield Enabled	✓
Grab Objects Enabled	✓
Change Camera View Enabled	✓
Shoot Enabled	✓
Change Powers Enabled	✓
Cursor	cursorGrabObjects
Run Mat	blackHole
Layer	Mixed ...
Trails Active	✓
Slider	slider (Slider)
Health Bar	healthBarSlider (Slider)
Power Bar	powerBarSlider (Slider)
Run Velocity	1.5
Run Jump Power	15
Run Air Speed	20
Run Air Control	4
Grab Radius	5.5
Joint Objects Force	40
▶ Able To Grab Tags	
Button Kick Texture	flatDark37
Button Shoot Texture	ico-7
Button Shoot	shoot
High Friction Material	Max Friction
Gravity Objects Layer	Mixed ...
Damage Icon	damageDirection
Max Alpha Damage	0.4

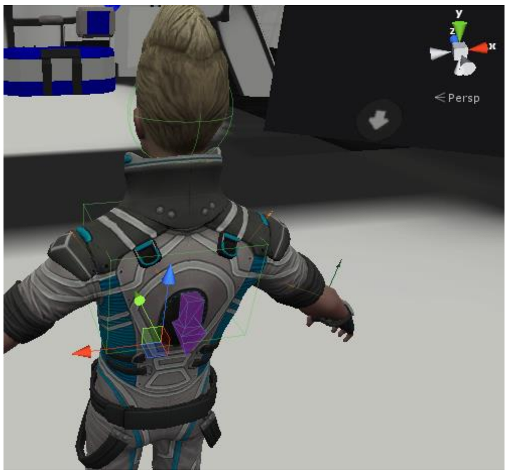
- The first 7 booleans are to enable or disable some features of the scrip, like run, aim mode, use shield, grab obejcts, change camera view, shoot, and change between powers. This features can also be changed in the features manager inspector in GTC_Prefab.
- **Cursor**, the UI image used as a cursor in aim mode.
- **Run mat**, the material applied to the player's model when he runs.
- **Layer**, a layermask used to the raycast used to shoot, and check other functions.
- **Trails active**, enable the trails renderer in the player when he runs.
- **Slider**, the slider showed when the player hold the grab objects button to throw them
- **Health and power bar**, the sliders used for the HUD.
- Run velocity, jump power, air speed and air control are the variables modified in playeController when the player runs.
- **Grab radius**, the radius distance used to grab objects close to the player.
- **Joint objects force**, the amount of force applied to the objects when the join power is used.

- **Able to grab tags**, the tags used in the objects that the player can grab, like boxes and friend turrets.
- **Button kick and shoot texture**, are the textures uses for the touch controls. Both textures are used in the same button, and are changed when the player is in regular or aim mode.
- **Button shoot**, the button that use both textures.
- **High friction material**, the material applied to the collider of an object that the player has changed its gravity.
- **Gravity objects layer**, the layermask set to an object with the gravity changed, to check if the objects is in the ground or in the air.
- **Damage icon**, the UI image used for the arrow used for the damage direction of the player.
- **Max alpha damage**, how much changes the alpha value of the red damage screen when the player is being damaged.

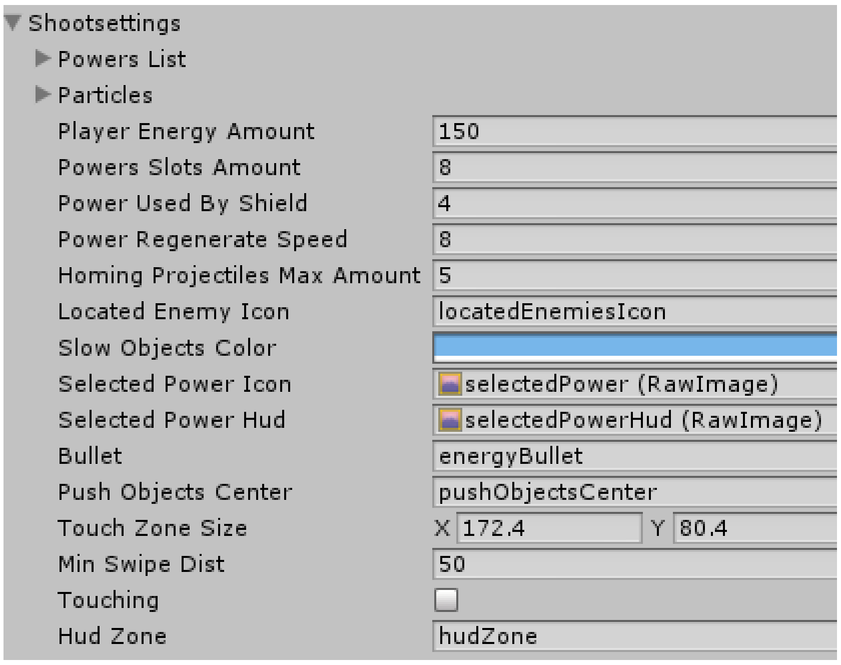

Aimsettings

Aiming	☐
Aim Side	Right
Spine	char_ethan_Spine2
Left Hand	char_ethan_LeftHand
Right Hand	char_ethan_RightHand
Spine Vector	X -1 Y 0 Z 0

- **Aiming**, a Boolean to show if the player is in aim mode or not.
- **Aim side**, an enum to set the side of the camera when the player aims.
- **Spine, left hand and right hand**, are used to store the IK goals in the player used for the aim mode, this fields are configured by the buildPlayer script when you press the button in its inspector, just change the spine in case that the bones in the player are not correctly, which is the chest of the model.
- **Spine vector**, since every skeleton has its own rotations and axis directions, set in this vector the correct X axis in the spine of your player, to rotate the body of the player in the camera direction in aim mode.
  In the image below, you can see how the Global X axis in the spine of the player aims in the negative X axis, so the rotation is made in that direction, being the vector (-1,0,0). Other models would have (0,0,-1) for example. Just check the spine axis in your model and changes this value



IMPORTANT: the configuration for version 2.3 has been changed, simplified and improved, for the current version (2.3.5.). Now the aim system uses Inverse kinematics for arms, so the only configuration needed is only place the red boxes positions where you want to make the player move his hands to that place in aim mode.

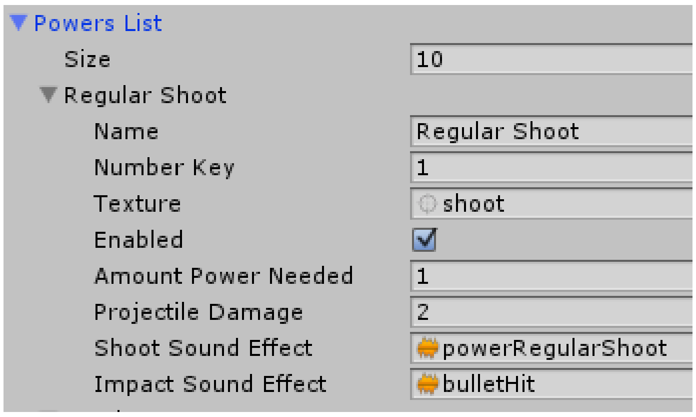


Shootsettings
- Powers List
- Particles

Player Energy Amount	150
Powers Slots Amount	8
Power Used By Shield	4
Power Regenerate Speed	8
Homing Projectiles Max Amount	5
Located Enemy Icon	locatedEnemiesIcon
Slow Objects Color	🟦
Selected Power Icon	selectedPower (RawImage)
Selected Power Hud	selectedPowerHud (RawImage)
Bullet	energyBullet
Push Objects Center	pushObjectsCenter
Touch Zone Size	X 172.4 Y 80.4
Min Swipe Dist	50
Touching	☐
Hud Zone	hudZone

- **Particles**, a list of particles used for the powers.
- **Player energy amount**, the initial amount of energy used for the powers.
- **Powers slots amount**, used for the amount of powers placed in the powers wheel, in the powers manager. Since the wheel has 8 powers, this is the limit of the current powers used by the player.
- **Power used by shield**, how much energy the shield uses along the time.
- **Power regenerate speed**, the speed that the player's energy is regenerated, if it is equal to 0, the energy is not regenerated.
- **Homing projectiles max amount**, the amount of enemies that can be locked at the same time using to shut down power.
- **Located enemy icon**, the icon used to locked the enemies using the homing projectiles.
- **Slow objects color**, the color applied to the materials of the objects that can be slowed, when the slow down power is used.
- **Selectd power icon and hud**, the UI image used when the payer changes between powers, to show the current power and the UI image in the upper left corner of the screen.
- **Bullet**, the prefab projectile that the player shoots.
- **Push objects center**, an empty gameobject inside the camera, used to get a radius center to store the objects close to the player when he uses the push objects power.
- **Touch zone size**, the size of a rect zone located in the upper left corner of the screen, used for the touch controls, when the player swipe to left or right touching inside that rect, the current power is changed.
- **Min swipe dist**, the minimum swipe distance to change between powers.

- **Hud zone**, the center of the touch zone size.



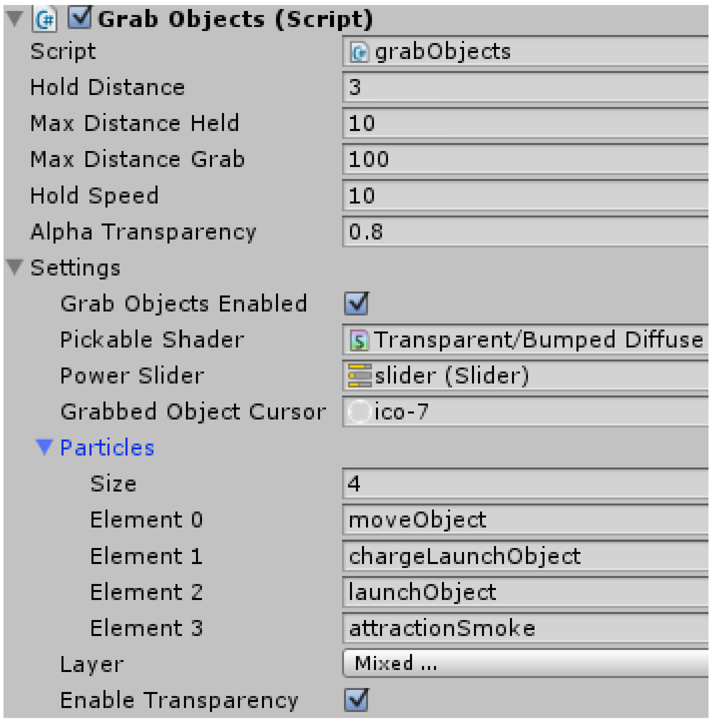
```
▼ Powers List
    Size                        10
  ▼ Regular Shoot
      Name                      Regular Shoot
      Number Key                1
      Texture                   shoot
      Enabled                   ✓
      Amount Power Needed       1
      Projectile Damage         2
      Shoot Sound Effect        powerRegularShoot
      Impact Sound Effect       bulletHit
```

- Inside shootSettings, there is a list to create new powers. If you want to add a new one, increase the size of the list, set a name to the power, the default key number (Uses from the 1 to 8 to use the slots in the wheel power, the rest is stored and ready to use by editing the current powers in the powers manager menu), the icon for that power, if the powers is enabled from the beginning of the game and the amount of power that uses.
- New attributes added are projectile damage, the amount of damage produced by that power to an enemy, the shoot sound effect for the clip audio played when the power is used and the sound played when the projectile hits a surface. The sounds effect can be add or not in the inspector.

GRAB OBJECTS, in the aim mode, you can look towards an object with a rigidbody and the tag box. Press E and the object will move in front of the player. To this, the objects is checked with a raycast, and its gravity is disabled, adding velocity to its rigidbody to move it in the camera direction. If the transparency value is higher than 0, the object will has a transparent shader, to see through it, changing the shader of all the objects materials inside it. The script will check the distance from the object to the camera, and if it is higher than a value, the object is dropped. There are different types of objects that can be grabbed:

- **Regular rigidbody**: it can be grabbed, dropped and thrown, with a force according to the time that the button has been pressed. Press one to grab. Press another time and drop. Hold and release to throw the object. In this mode, a slider is visible showing the total amount of force.
- **Rail object**: the object can be moved in two directions, in the forward and back direction of the object. The object has a limit position, to prevent the object move too far and a final position, where the object is engaged. To check in what direction the object has to be moved, it is used the angle between the player and the object, so when its angle is lower or higher that a value, the objects displaces, else the objects still in its place.
- **Rotate object**: in this case, the object is rotate, in the Z axis to activate another device.
- **Friend turret**: when a turret is hacked, it can be grabbed to change its position.
- **Barrel**: if a barrel is grabbed, a function is called to disable the barrel from explode by a collision until the player drops or throws it.

- **Vehicles**: when the raycast detects a vehicle damage receiver, it gets the vehicle that uses it and the vehicles is set as object to held, so like any other object, the vehicle can be dropped, carried, thrown or change its gravity pressing the right mouse button in any direction camera.



Grab Objects (Script)

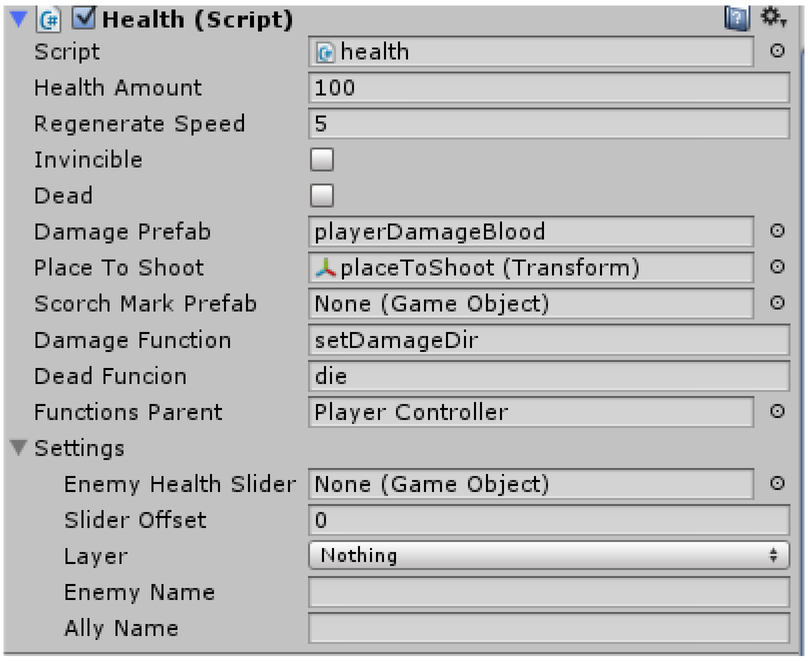
Script	grabObjects
Hold Distance	3
Max Distance Held	10
Max Distance Grab	100
Hold Speed	10
Alpha Transparency	0.8
Settings	
Grab Objects Enabled	✓
Pickable Shader	Transparent/Bumped Diffuse
Power Slider	slider (Slider)
Grabbed Object Cursor	ico-7
Particles	
Size	4
Element 0	moveObject
Element 1	chargeLaunchObject
Element 2	launchObject
Element 3	attractionSmoke
Layer	Mixed ...
Enable Transparency	✓

- **Hold distance**, the distance from the player to the object when it is grabbed.
- **Max distance held**, if the object is blocked by a wall or other element, if the distance between the object and the player is higher than that amount, the object is dropped.
- **Max distante grab**, the maximum distance where the object can be grabbed.
- **Hold speed**, how fast the object is moved when the player moves it.
- **Alpha transparency**, if the transparency is enabled, the alpha value of the material in the object, to see through it.
- **Pickable shader**, the shader used for the transparency, applied to all the materials in the grabbed object.
- **Power slider**, the slider showed when the player is going to throw the object.
- **Grabbed object cursor**, the icon used for the cursor in the center of the screen when the player grabs an object.
- **Particles**, a list of particles system used for this ability.
- **Layer**, a layermask used for the grabbing raycast.
- **Enable transparency**, if true, set a transparent shader in the materials of the grabbed object.

HEALTH, allows that every character, enemy, npc, etc, in the game has an amount of life. The amount of life can be limited or infinite. When an object with the health component is damaged, a particles system is activated. Also has a shoot position to be fired, to avoid that any enemy shoot in the feet of the player, due to its global position is in his feet, so like this, the enemies will shoot to the center of the player, or any other character. A scorch can be placed in the position where the enemy dies.

There are two functions to be called to the owner of the script: damage function and die function. To every object with health, the functions can be different, for example, when the player is damaged, the screen fades to red and an icon show the damage direction, and when the player dies, the function activates the ragdoll and disable the controls. When a turret is damaged, this search the owner of the bullet that damaged it, to shoot him. And when the turret dies, its textures fade and a rigidbody and a force are added to its renderer objects, to finally destroy the turret object.

Finally, to any object different to the player, the health component has a slider which is instantiated to set it above the object, to show its life and its name. Every object with health, will be added in the radar of the player, to be visible in it.



Health (Script)

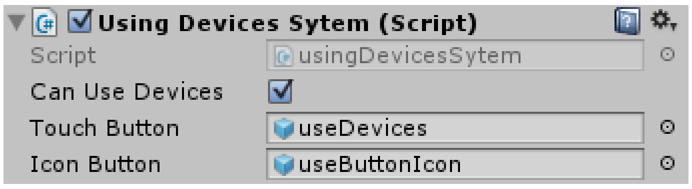
Script	health
Health Amount	100
Regenerate Speed	5
Invincible	☐
Dead	☐
Damage Prefab	playerDamageBlood
Place To Shoot	placeToShoot (Transform)
Scorch Mark Prefab	None (Game Object)
Damage Function	setDamageDir
Dead Funcion	die
Functions Parent	Player Controller
Settings	
Enemy Health Slider	None (Game Object)
Slider Offset	0
Layer	Nothing
Enemy Name	
Ally Name	

- **Health amount**, the initial amount of health for every character.
- **Regenerate speed**, the speed what the health is regenerated. If the value is equal to 0, the health is not regenerated.
- **Invincible**, if true, the character is not damaged.
- **Dead**, the character has died.
- **Damage prefab**, a particle emitter used when the character is damaged.
- **Place to shoot**, the position where the NPCs aim when they shoot.
- **Scorch Mark Prefab**, the scorch that the character set in the ground when it is destroyed.
- **Damage and dead function**, string functions name called when the character is damaged or dies. They can be used for example to call the death functions in player or an enemy, or for barrels, so when they explode, they call the drop pickups component.
- **Functions parent**, the object to send the above functions message.
- **Enemy Health slider**, a slider prefab to set above NPCs, to see their names and their health bars.
- **Slider offset**, an offset for the bars sliders.
- **Layer**, a layermask used to check their ground direction to set the scorch when the character dies. Also in the NCPs is used to make visible their bar sliders when the player can see the character, using raycast and distance.

- **Enemy name**, if the NPC is an enemy, set their name showed in the bar above it.
- **Ally name**, else the NPC is an ally, so set its name in the bar.

USING DEVICES SYSTEM, it allows to use any device in the game, like computers, text device, code panels, buttons, get in and off from vehicles, use doors, etc …. Also, it gets the string action configured in a device and show it in the icon screen when the player is inside a device trigger. This script only detects a device using the tag device in a trigger.

Also, the device needs an interaction info, configured in the script **deviceStringAction**, showing or not the icon in the screen, disabling the icon when the device is active or not and showing or not the touch button of use a device.


Using Devices Sytem (Script)

Script	usingDevicesSytem
Can Use Devices	✓
Touch Button	useDevices
Icon Button	useButtonIcon

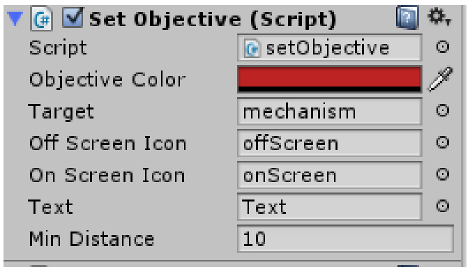
- **Touch button**, the touch button used to enable any device in the game, placed in the touch controls. Since this button is only visible when the player is inside a device trigger, the button is disable the rest of the time.
- **Icon button**, the icon placed in a device position in the screen to show the player that can use that device.

SET OBJECTIVE, allows to set an icon in the screen which follows the position of a target that the player has to reach to use it. If the objects is visible in the screen, the icon show the distance to the target, else the icon is an arrow which follows the object moving in the corner of the screen.

This script also change the color of the target materials. When the distance between the player and the target is lower than a certain amount, the icon is disabled, and the color of the materials are set to their previous color.

To set a target, just add the object in the target field, or call this function:

Player.GetComponent<setObjective>().target=yourObject

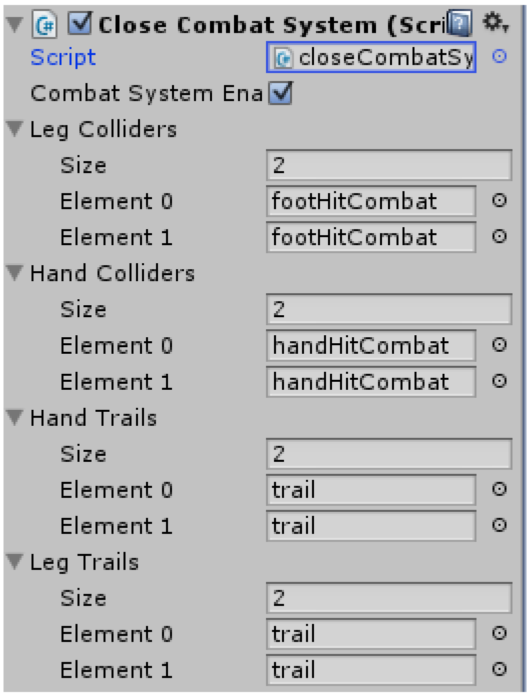

Set Objective (Script)

Script	setObjective
Objective Color	🟥
Target	mechanism
Off Screen Icon	offScreen
On Screen Icon	onScreen
Text	Text
Min Distance	10

- **Objective color**, the color applied to all the materials of the objective color, to make it more visible to the player.
- **Target**, the current objective, it can be set in the inspector or by code.

- **Off screen icon**, the icon showed when the player is not looking in the objective direction.
- **On screen icon**, the icon showed when the player is looking in the objective direction.
- **Text**, the text showed in the icon screen to see the distance from the object to the player.
- **Min distance**, when the distance between the player and the objective is lower than the value, the objective is reached, the icon in the screen is removed and the color materials of the object are back to the regular state.

CLOSE COMBAT SYSTEM, it allows the player to use combos of kicks and punches, setting the parameters in the combat layer of the animation. Also it enables and disables the colliders and trail renderers in the hands and feet of the player, according to the type of combat. The left mouse button activates the punches and the right the kicks.



Close Combat System (Scri

Script	closeCombatSy
Combat System Ena	✓
Leg Colliders	
Size	2
Element 0	footHitCombat
Element 1	footHitCombat
Hand Colliders	
Size	2
Element 0	handHitCombat
Element 1	handHitCombat
Hand Trails	
Size	2
Element 0	trail
Element 1	trail
Leg Trails	
Size	2
Element 0	trail
Element 1	trail

- **Leg and hand colliders**, these are the triggers in the hands and feet of the player, used to check if the player touches an enemy when he uses the close combat.
- **Hand and leg trails**, these are the trails renderer used when the player runs, used also when the player is fighting, just to add a nice effect to the game.
- **These elements are set automatically for the build player function, so it doesn't need any configuration.**

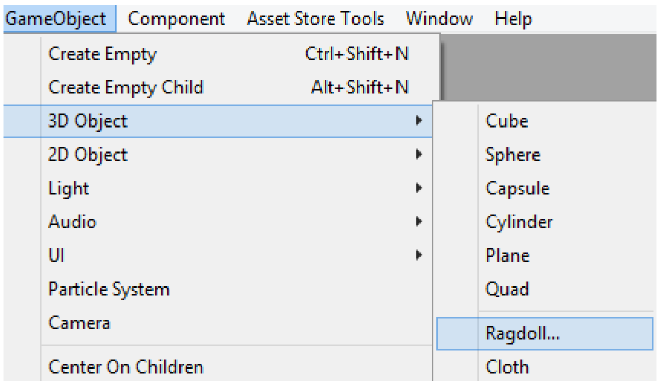
RAGDOLL ACTIVATOR, when the player dies, all the rigidbodies and colliders inside the character body, are enabled, the animator is disabled and other script functions are disabled, to prevent the player can play while the character still dead. Before this, all the bones rotations and positions are stored.

When the player press the menu button to play again, the bones are rotated and moved to the position of the get up animation, to get a smooth transition from the ragdoll to the animator. Also, it is check if the player is above his belly or his back, by checking the rotation of the hips.

Finally, the colliders and rigidbodies inside the character are disabled.

IMPORTANT: if you want to use other model to the player and use a ragdoll when the player dies, you have two options:

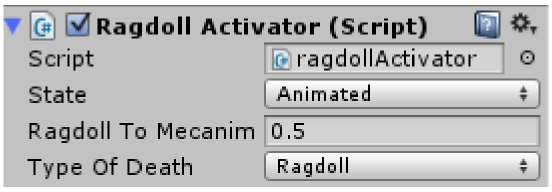
- **Create a ragdoll with the wizard menu of unity**, setting every bone in the correct field. The script will detect if the character has a ragdoll or not.



- **Use the ragdollBuilder inspector in the Character gameObject**, pressing build ragdoll button, making a procedural ragdoll very similar to the one made by the wizard.

A new function has been added, to use ragdoll or animations when the player dies. Also, if you set your own character and you don't add a ragdoll to it, the script will use animations instead ragdoll, until you add it to the player.

It is not mandatory to do this to be able to use this asset, just in case you want to use a ragdoll instead a death animation.


Ragdoll Activator (Script)

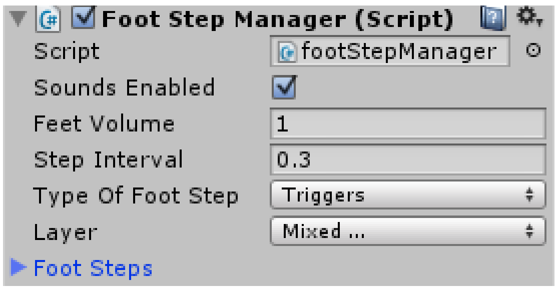
Script	ragdollActivator
State	Animated
Ragdoll To Mecanim	0.5
Type Of Death	Ragdoll

- **State**, show the current state of the player, animated when he uses the animator component, ragdolled when the player has died and blend to anim when the player is getting up.
- **Ragdoll to Mecanim**, is the speed of rotation of the bones when the player gets up.
- **Type of death**, set if you want to use ragdoll or animations when the player dies.

FOOTSTEPS MANAGER AND **FOOTSTEP**, the footsteps manager uses the sphere collider triggers in the feet of the player, which both have the footstep script, to play one shoot of the footstep sound, every time it collides in any surface, using the animation of the player to touch any surface. Also it checks the layer of the mesh object, to play a sound according to that layer. If the player is walking in a terrain, the manager checks the index of the current texture, searching if that texture is defined in the footsteps list, to play a sound according to the type of texture. In the current asset there are 5 different sounds and surfaces for mesh objects:

- Regular, the normal footstep sound used in the default layer.
- Metal
- Snow
- Water
- Wood

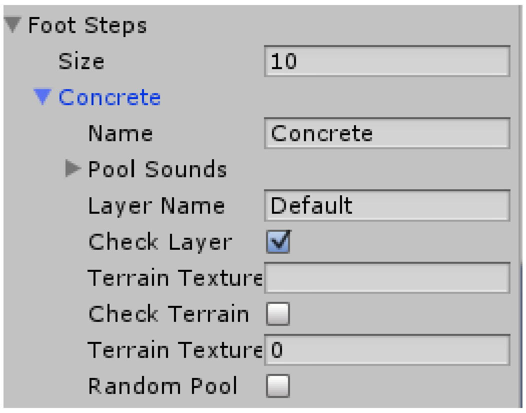
For terrains, you can define any type of new surface.



Foot Step Manager (Script)

Script	footStepManager
Sounds Enabled	✓
Feet Volume	1
Step Interval	0.3
Type Of Foot Step	Triggers
Layer	Mixed ...
Foot Steps	

- **Sounds enabled**, the footsteps are calculated and played.
- **Feet volume**, the volume of the audio source in the feet.
- **Step interval**, if the type of foot step is raycast, this value set the frequency of the footsteps.
- **Type of foot step**, you can choose if the footsteps are calculated with the sphere collider triggers in the feet of the player, or using a raycast towards the surface below the player. Both options use the same functions and system, the only difference is the accuracy of the footsteps, being better to use triggers, but you can choose the option that you prefer.
- **Foot steps**, is the list that contains every type of surface defined. You can add your own sounds and new layers easily, to have more variety. There are two types of objects for the footsteps:
    - **Mesh object**, if any 3D object in the scene, cubes, spheres, models, … To check this type of objects, the system uses layers. You have the previous 5 layers already define, but you can define and set any new layer to add a new type of surface.

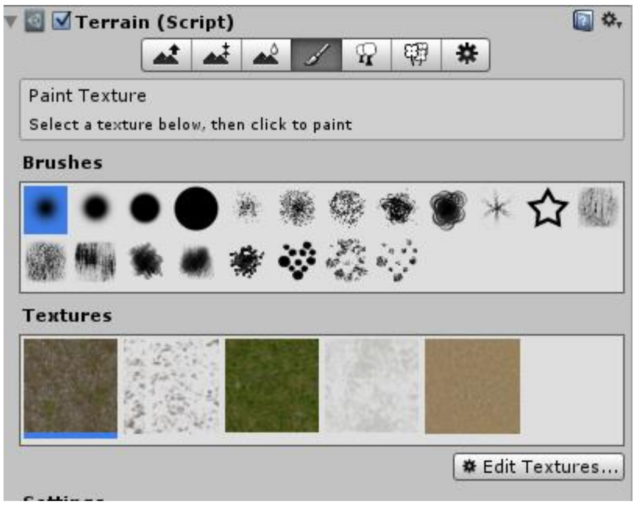


Foot Steps

Size	10
Concrete	
Name	Concrete
Pool Sounds	
Layer Name	Default
Check Layer	✓
Terrain Texture	
Check Terrain	☐
Terrain Texture	0
Random Pool	☐

If you want to define a new surface of this type, increase the size of the list, set a name, a pool of sounds, it can has 1 or more sounds, press check layer and finally set to true or false random pool if you want to use the list of sounds randomly or in the order that you have added them.

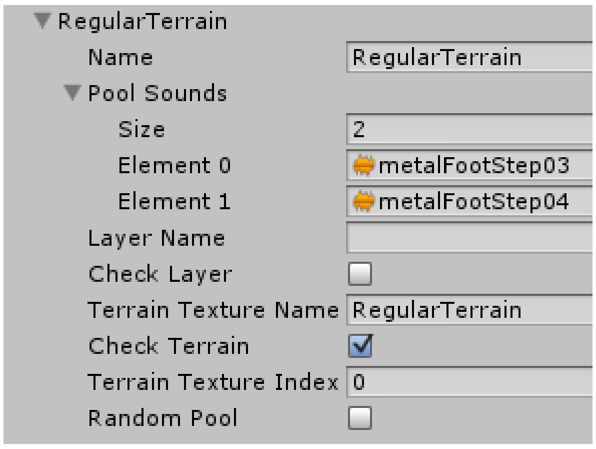
- **Terrains**, for this object, the system uses the index of the textures added to the terrain. In the below image, you can see that these terrain has 5 textures.



To add the first texture to the footsteps system, add a new element to the list of footsteps, set a name, the pool of sounds, the name of the footstep, select check terrain and in terrain texture index set 0, as the first texture added to the terrain. Finally set if the sounds are played randomly or not.

For the next terrain texture, set the same configuration, but now the terrain texture index will be 1. And like this with the new terrain footsteps.

For the next version this configuration will be improve, adding the texture directly to the inspector instead using the index.



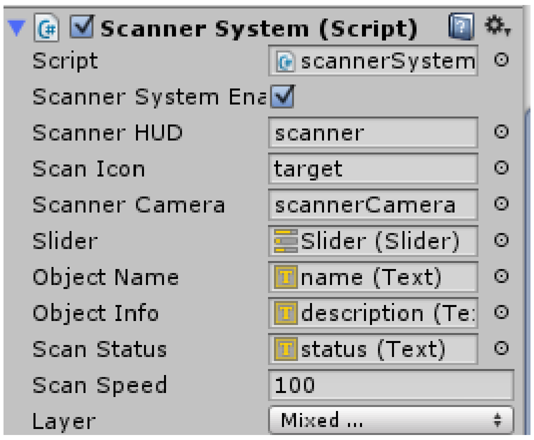
RegularTerrain

Name	RegularTerrain
Pool Sounds	
Size	2
Element 0	metalFootStep03
Element 1	metalFootStep04
Layer Name	
Check Layer	☐
Terrain Texture Name	RegularTerrain
Check Terrain	✓
Terrain Texture Index	0
Random Pool	☐

SCANNER SYSTEM, this system is used to add hidden info for any object in the game, so you can set a little name and description to an object like an important element, an enemy, an ally, etc… To this, when the player enables the scanner mode, if the camera is in third person, it is changed to first, else the camera is not changed.

Then a little camera inside the main camera is enabled, showing models above the scannable object in a layer that the main camera doesn't see, using layers, so the player can see clearly what objects can be scanned. These models are just a 3d model with a **SCAN ELEMENT INFO** component and a trigger, so when the player looks at that object, the raycast detects it.

If the object hasn't be scanned, the player has to hold the scanner button until the object is scanned. Then the info of that object is showed. Now if the player press again the scanner button, the info is showed again without any waiting.



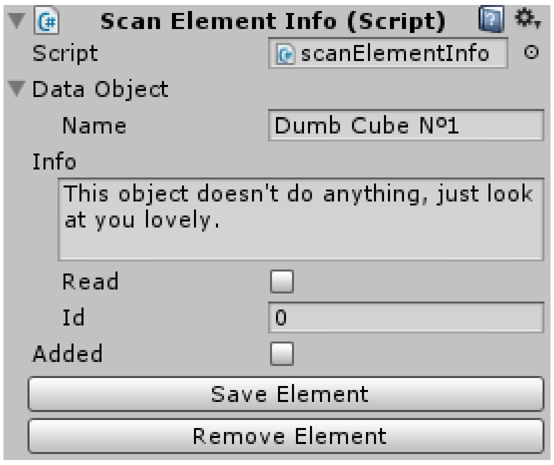
Scanner System (Script)

Script	scannerSystem
Scanner System Ena	✓
Scanner HUD	scanner
Scan Icon	target
Scanner Camera	scannerCamera
Slider	Slider (Slider)
Object Name	name (Text)
Object Info	description (Te:
Scan Status	status (Text)
Scan Speed	100
Layer	Mixed ...

- **Scanned hud**, is the part of the hud that contains all the elements of the scanner visor mode.
- **Scan Icon**, it is the green cursor that follows a locked scannable object when the player looks at one of them.
- **Scanner camera**, the little camera showed in the center of the screen to show the models placed above an scannable object, to make more visible all the objects that can be scanned.
- **Slider**, is the slider used to "scan" the object.
- **Object name**, the text component where the name of the object scanned is showed.
- **Object info**, the text component where the description of the object scanned is showed.
- **Scan status**, a text to show the current state of the scanner.
- **Scan speed**, the speed to scan an object.
- **Layer**, the layermask used to check the scannable objects trigger.

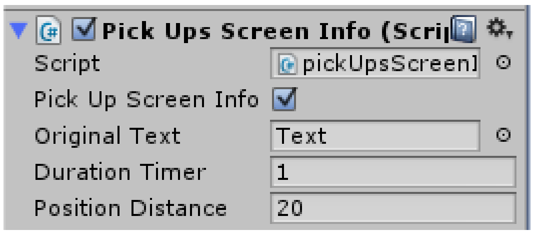
SCAN ELEMENT INFO, store the info of a scannable object. It has a name and a description, showed when the player scans the object. Once that the player has scanned it, the read Boolean is set to true, so the player hasn't to scan again to see the info one more time.

Right now, the script has a function to store every object info in a file, so you can make a database of scannable objects, saving or removing every element added. But this system is not fully ready, tough it already works, but there isn't a menu so the player can see the database of objects scanned by him. But for the version 2.4, this system will be completed, added for an interactive menu with inventory, missions and this database.


Scan Element Info (Script)
Script scanElementInfo
Data Object
Name Dumb Cube Nº1
Info
This object doesn't do anything, just look at you lovely.
Read ☐
Id 0
Added ☐

- **Name**, the name of the scannable object.
- **Info**, the description of that object.
- **Read**, if true, the object has been scanned by the player.
- **Id**, an auto incremental integer value, to search easily every object info in the database.
- **Added**, if true, the object has been saved in the database.
- **Save element**, store the object info in the database file.
- **Remove element**, remove this object info from the database file.

PICK UP INFO SCREEN, a system to show the name and the amount that add to the player when a pickable object is grabbed by the player. This script is called by pickable object grabbed by the player. If the player grabs a health pickup, which restore 30 units of health, in the screen is showed the text: "Health x30".

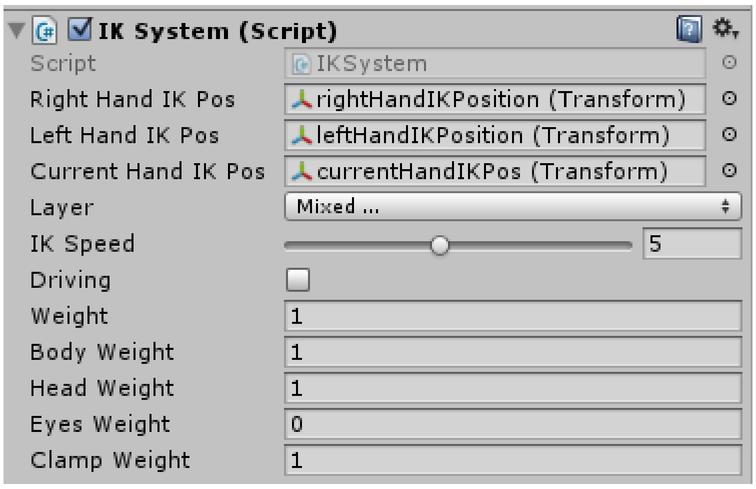

Pick Ups Screen Info (Scrip
Script pickUpsScreenI
Pick Up Screen Info ✓
Original Text Text
Duration Timer 1
Position Distance 20

- **Text**, a text component, placed in the right middle side of the screen. It is used as the element to clone, to show many text element, to show the list of objects grabbed by the player. It is also used as reference for the position where the text will be placed, setting new text above the original one.
- **Duration timer**, the amount of time before every text showed is disabled.
- **Position distance**, the distance between a text and the next above it.

IK SYSTEM, it is used for two modes:

- **When the player aims**, it set the current hand position and rotation used to aim to the position and rotation configured in the red boxes of the player, setting the IK goals from the player's bones (hands) in the correct position. Like this the player can move his arm without animations and it can be configured easily where the player will place his hands.

- **When player drives**, getting a list of configured positions from vehicles **IK Driving System** component to set every limb part of the player correctly to drive that vehicle and move the player's arms according to the steering wheel.

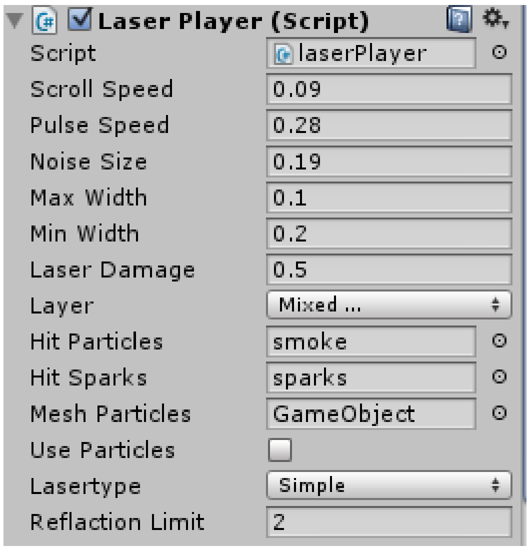


IK System (Script)

Script	IKSystem
Right Hand IK Pos	rightHandIKPosition (Transform)
Left Hand IK Pos	leftHandIKPosition (Transform)
Current Hand IK Pos	currentHandIKPos (Transform)
Layer	Mixed ...
IK Speed	5
Driving	☐
Weight	1
Body Weight	1
Head Weight	1
Eyes Weight	0
Clamp Weight	1

- **Right hand IK pos** and **left hand IK pos**, they are the red boxes in the player's body used to set where the hands of the player will be placed and rotated when he aims.
- **Current hand IK pos**, the current transform position where the current hand is aiming in the aim mode.
- **Layer**, the layer used to check for colliders in front of the player when he is aiming, so according to the distance from the hand to a wall, the hand is moved backwards to avoid cross the collider.
- **Ik speed**, the velocity which the hand is moved from its animation position to the current hand IK position.
- **Driving**, if true, the player is driving.
- **Weights**, used in motorbike, setting the weight of the IK in the player's body parts.

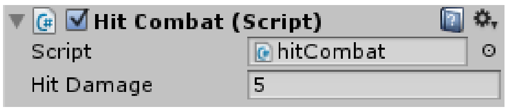
LASER PLAYER, when the player touches a laser device and enables the aim mode, a laser is activated in the hand of the player, projected in the camera direction. This laser is used to activate other devices, using also a refraction cube to deflect the laser, changing its color, to make puzzles for example. The laser can use as weapon to damage any object with a health or a vehicle damage receiver component. A new function has been added to reflect the laser in any surface. To get this, the player has to touch a laser device that allow to reflect the player's laser. In this mode, the last reflected laser is the one that checks if the laser is touching a laser receiver, to connect a device, or an enemy, to damage it.


Laser Player (Script)

Script	laserPlayer
Scroll Speed	0.09
Pulse Speed	0.28
Noise Size	0.19
Max Width	0.1
Min Width	0.2
Laser Damage	0.5
Layer	Mixed ...
Hit Particles	smoke
Hit Sparks	sparks
Mesh Particles	GameObject
Use Particles	☐
Lasertype	Simple
Reflaction Limit	2

- **Scroll speed, pulse speed, noise size and max and min width** are used to animate the laser, set the values as you want.
- **Laser damage**, is the amount of damage that the laser makes to any character with a health component.
- **Layer**, is the layermask used to check laser receivers, refractions cubes or enemies, and to check collisions with the rest of colliders.
- **Hit particles and hit sparks**, used for the smoke and sparks particles placed in the point where the laser hits a surface.
- **Mesh particles**, used for a particles emitter to add a nice effect to the player. It will be improved in the next version.
- **Use particles**, enable or disable the mesh particles gameObject.
- **Lasertype**, this is configured by a laser device that touches the player, but it can be configured set if the laser has or not reflections..
- **Refraction limit**, used to set the amount of reflections that the laser does. The minimum amount of refractions is 2.

HIT COMBAT, used for the sphere collider triggers in the hands and feet of the player, to check if the player touches an object with a health component when he uses the close combat system, damaging that object.

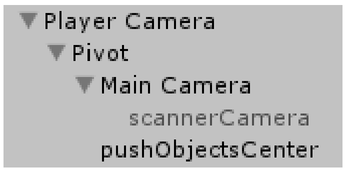

Hit Combat (Script)

Script	hitCombat
Hit Damage	5

- **Hit damage**, used to set the damage produced in every limb.

PLAYER CAMERA

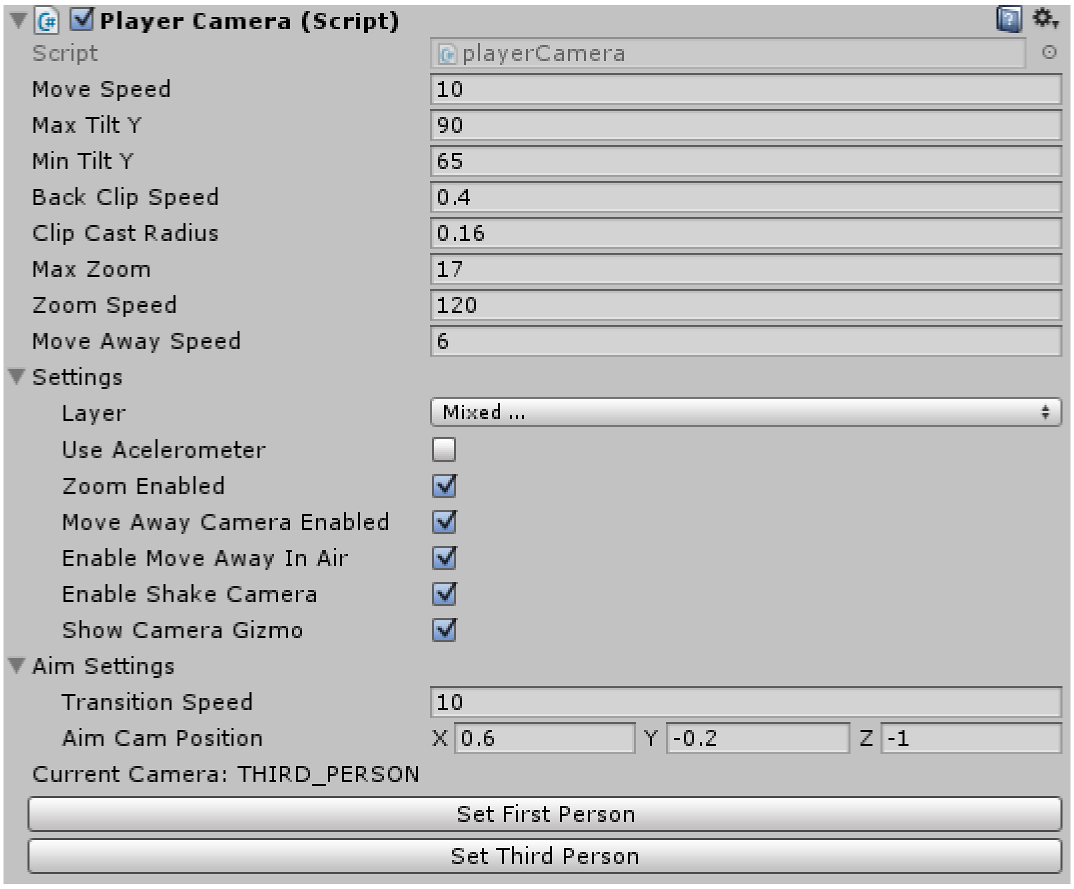
Hierarchy order, it contains the pivot of the player's camera and the pivot of the main camera, inside of it. The scanner camera is only enabled when the scanner mode is active. Push objects center is used as a position reference to push objects inside a radius, using that empty gameObject as the center of that radius.


Player Camera
Pivot
Main Camera
scannerCamera
pushObjectsCenter

PLAYER CAMERA, take the values of the mouse, to rotate the camera. Also it moves its position to avoid cross any collider close to the player, using a sphere cast. It has the right joystick to move the camera, and some parameters to set the first and third camera view.

In the version 2.0, it moves the camera from its regular position towards the shoulder of the player. Also allows to change between right and left side of the player to aim. Another feature is a function to use a zoom mode, and to move away the camera, so the player can see better the scene around him, by elevating the camera above the player. Finally, the camera position is modified when the player crouches.

A new function has been added to stop the camera rotation when the player use a device which moves the camera position or he is driving.


Player Camera (Script)

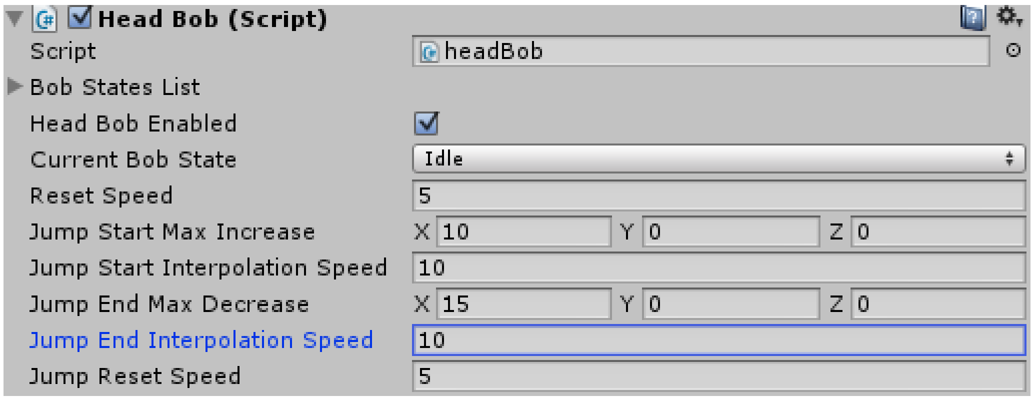
Script	playerCamera
Move Speed	10
Max Tilt Y	90
Min Tilt Y	65
Back Clip Speed	0.4
Clip Cast Radius	0.16
Max Zoom	17
Zoom Speed	120
Move Away Speed	6
Settings	
Layer	Mixed ...
Use Acelerometer	☐
Zoom Enabled	✓
Move Away Camera Enabled	✓
Enable Move Away In Air	✓
Enable Shake Camera	✓
Show Camera Gizmo	✓
Aim Settings	
Transition Speed	10
Aim Cam Position	X 0.6 Y -0.2 Z -1

Current Camera: THIRD_PERSON

- **Move speed**, a multiplier to adjust the rotation speed of the camera.
- **Max tilt Y**, the max positive tilt rotation to look above the player.
- **Min tilt Y**, the min negative tilt rotation to look below the player.
- **Back clip speed**, the velocity which the camera is moved when collides with a surface, to move it from its position to another, to avoid cross a surface collider.

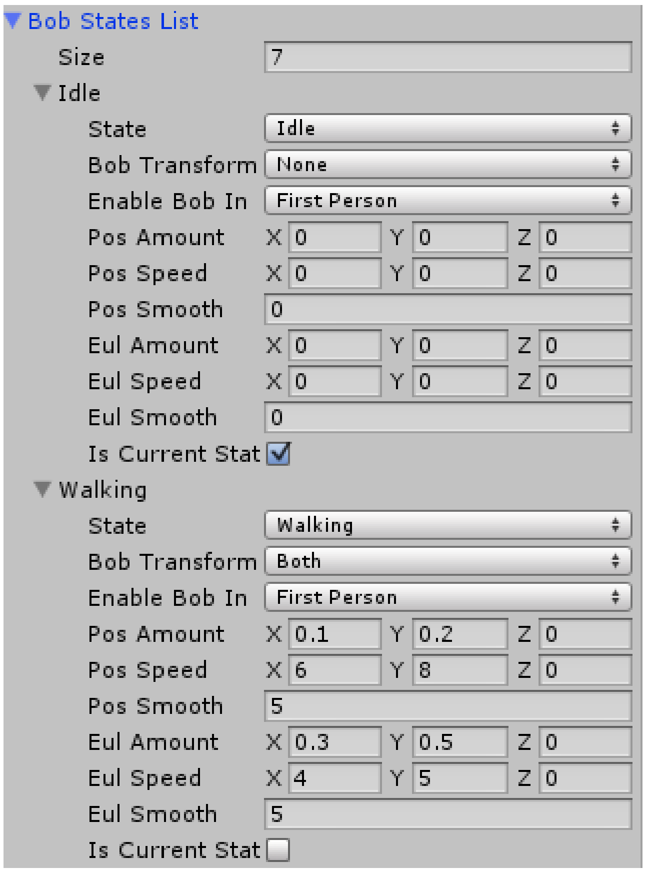
- **Clip cast radius**, the radius used to detect collisions in the camera with any object.
- **Max zoom**, the fov value set in the camera in zoom mode.
- **Zoom speed**, the velocity to change the fov of the camera.
- **Move away speed**, the velocity of the camera when is moved away from its original position.
- **Layer**, the layermask used for the raycast to calculate the collision in the right, left and back side of the camera when the player is in aim mode.
- **Use accelerometer**, if true, when the game is played in a touch device and the player is in aim mode or selecting a gravity direction, an extra rotation is applied to the camera, according to the rotation of the device.
- **Zoom enabled**, if true, the player can use the zoom mode.
- **Move away camera enabled**, if true, the player can move away the camera from the player.
- **Enable move away in air**, if true, when the player accelerates his movement in the air, the camera fov is changed.
- **Enable shake camera**, if true, when the player accelerates his movement in the air, or he falls and reaches a certain velocity, or when the player searches a surface, the camera is shaken using the headbob system in the main camera.
- **Show camera gizmo**, if true, in editor mode you can see a draw line in the player camera.
- **Transition speed**, the velocity of the camera moving from regular mode to aim mode and viceversa.
- **Aim cam position**, the local position where the camera is placed in aim mode.
- **Current camera**, a label text to show the current camera view of the player.
- **Set first person**, if the player is in third person mode, the camera is changed to a first person view in editor move, so you can start the game in first person mode.
- **Set third person**, if the player is in first person mode, the camera is changed to a third person view in editor move, so you can start the game in third person mode.

HEAD BOB, a headbob system used to shake the camera when the player is in a certain state, like walking running, moving in the air, etc.… It allows to define different headbob states, used when the player make a certain action. Also, it allows to define the jumping and landing shake. If you add new states, you will need also add the code according to the new states that you have added.


Head Bob (Script)

Script	headBob
Bob States List	
Head Bob Enabled	✓
Current Bob State	Idle
Reset Speed	5
Jump Start Max Increase	X 10 Y 0 Z 0
Jump Start Interpolation Speed	10
Jump End Max Decrease	X 15 Y 0 Z 0
Jump End Interpolation Speed	10
Jump Reset Speed	5

- **Bob states list**, is the list of headbob states, applied to the player in different situations. It contains the next values to every shaking state:
  - **State**, an enum to set a name for the shaking,
  - **Bob transform**, to set if the shaking translates, rotates or do both actions to the camera, and also don't make any action to the camera.
  - **Enabled bob in**, it is to set if the shake state is applied when the camera is in first or third person or both.
  - **Pos amount, speed and smooth**, used for the translate shaking, to set the distance applied to the local position, the speed of the movement and the smooth of it.
  - **Eul amount, speed and smooth**, used for the rotation shaking, to set the degrees applied to the local rotation, the speed of the rotation and the smooth of it.
  - **Is current state**, used to get the current shaking state to apply to the camera.


Bob States List

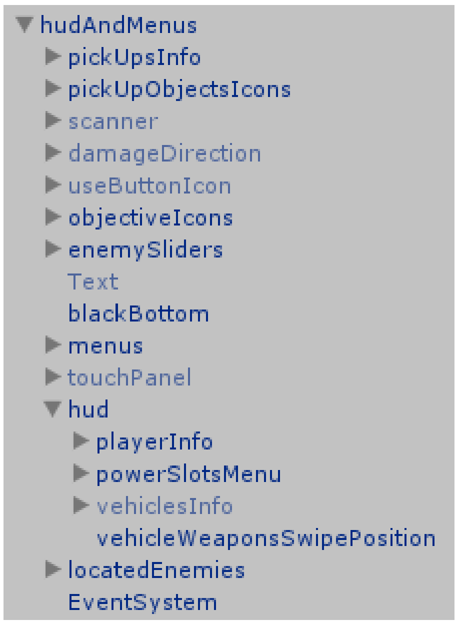
Size	7
Idle	
State	Idle
Bob Transform	None
Enable Bob In	First Person
Pos Amount	X 0 Y 0 Z 0
Pos Speed	X 0 Y 0 Z 0
Pos Smooth	0
Eul Amount	X 0 Y 0 Z 0
Eul Speed	X 0 Y 0 Z 0
Eul Smooth	0
Is Current Stat	✓
Walking	
State	Walking
Bob Transform	Both
Enable Bob In	First Person
Pos Amount	X 0.1 Y 0.2 Z 0
Pos Speed	X 6 Y 8 Z 0
Pos Smooth	5
Eul Amount	X 0.3 Y 0.5 Z 0
Eul Speed	X 4 Y 5 Z 0
Eul Smooth	5
Is Current Stat	☐

- **Head bob enabled**, if true, the camera is shaken according to its state.
- **Current bob state**, the current state of the player.
- **Reset speed**, the speed which the camera rotation and position is reset when the camera is not shaken.
- **Jump start max increase**, the degrees applied to the camera when the player jumps.
- **Jump start interpolation speed**, the speed of the jump rotation
- **Jump end max decrease**, the degrees applied to the camera when the player lands.
- **Jump end interpolation speed**, the speed of the land rotation.
- **Jump reset speed**, the speed used to reset the camera to the original rotation when the landing ends.

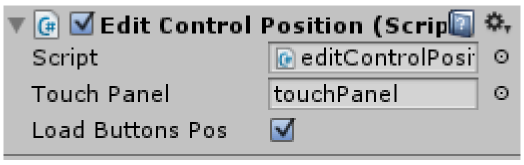
HUD AND MENUS

Hierarchy order, it contains all the elements used for the HUD, like menus, icons, buttons, etc…:

- **Pickups info**, used to show the info about the pickups grabbed. It contains the prefab used for all the pickup objects.
- **Pick up objects icons**, used to add icons to every pickup object, to show its type.
- **Scanner**, used for all the elements of the scanner visor.
- **Damage direction**, used for the arrow and the red fading screen when the player is damaged.
- **Use button icon**, used when the player is inside the trigger of a usable device.
- **Objective icons**, used for the objective icon that show the target position to the player-
- **Enemy sliders**, it contains all the characters health slider, showed above them.
- **Text**, just a text to show the message: Press esc for menu.
- **Black bottom**, an image panel to fade the screen when the game is paused.
- **Menus**, it contains all the menus of the game, like pause, edit input, etc…
- **Touch panel**¸ it contains all the touch buttons used for a mobile platform.
- **Hud**, it contains the elements of the HUD, like health and energy sliders, the current power selected, the powers manager and the cursor.
  Also, it contains the vehicle HUD with health, energy and weapon info from the current vehicle. It is only enable when the player is driving.
- **Located enemies**, used to set locked icons to the enemies using the homing projectiles power.
- **Event system**, the event system of the canvas component.

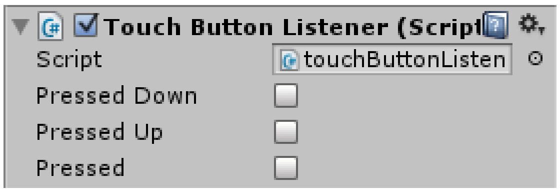

```
▼ hudAndMenus
  ▶ pickUpsInfo
  ▶ pickUpObjectsIcons
  ▶ scanner
  ▶ damageDirection
  ▶ useButtonIcon
  ▶ objectiveIcons
  ▶ enemySliders
    Text
    blackBottom
  ▶ menus
  ▶ touchPanel
  ▼ hud
    ▶ playerInfo
    ▶ powerSlotsMenu
    ▶ vehiclesInfo
      vehicleWeaponsSwipePosition
  ▶ locatedEnemies
    EventSystem
```

EDIT CONTROL POSITION, it allows to edit the control buttons in editor mode and touch devices and save their positions in a file, so the configurations are loaded in the next execution. Use the function to convert the mouse into a finger from touchJoystick. The load of the file at the start can be disable in its editor.


Edit Control Position (Scrip

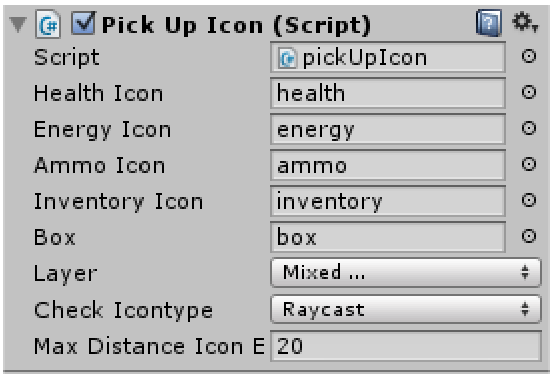
Script	editControlPosi
Touch Panel	touchPanel
Load Buttons Pos	✓

TOUCH BUTTON LISTENER, just a script to check if the UI buttons of unity 4.6 are being pressing, holding or released, to use repeat buttons of the old GUI system in the new UI. They are used for the touch controls, which at the same time is checked by the input manager, which checks the current value of the touch button listener used. This script has to be added to every touch button, instead of use event triggers and it doesn't need any configuration.


Touch Button Listener (Script

Script	touchButtonListen
Pressed Down	☐
Pressed Up	☐
Pressed	☐

PICKUP ICON, allows to set an icon that follows every pickup object in the game, to show the type of object, like health, energy, ammo, etc… When the pickup is grabbed, the icon is removed. Ammo and inventory pickups will be added in the version 2.4



Pick Up Icon (Script)

Script	pickUpIcon
Health Icon	health
Energy Icon	energy
Ammo Icon	ammo
Inventory Icon	inventory
Box	box
Layer	Mixed ...
Check Icontype	Raycast
Max Distance Icon E	20

- **Health, energy, ammo and inventory icon** are the icons used to every type of pickup object.
- **Box**, the other icon that follows the pickup.
- **Layer**, the layermask used to check if the icon is visible to the main camera or not.
- **Check icon type**, an enum to select if every icon of the pickup objects are visible using a raycast from the pickup to the player's camera, if there isn't anything until the ray reaches the camera, then the icon is visible, else the icon is disabled.

  In this mode, also the distance is checked, so even if the raycast reaches the camera, if the distance from the pickup to the camera is higher that max distance icon enabled, the icon is disabled.

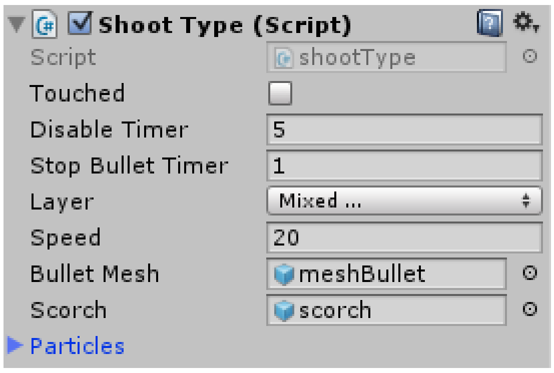
  Other option is used distance, so if the distance from the pickup to the camera is higher that max distance icon enabled, the icon is disabled.

  The final option is not use any of the previous methods, so if the player looks in the pickup direction, the icon is visible, not matter the distance or if any object is between the player and the pickup.
- **Max distance icon enabled**, the max distance from the pickup to the player before the icon is disabled if the check icon type used is raycast or distance.

OTHERS SCRIPTS

SHOOT TYPE, is the script that use the bullets fired by the player. It gets the power chooses in otherPowers, and make an action according to that power. There are different actions:

- Simple bullet: damages an object with a health component when it collides with that object. Has a system particles and a scorch which are set in the collision point.
- Grenade: damages every object with a health component inside a radius of action. Like the bullet, has a particle system and a scorch which are activated in the collision.
- Implosion grenade: attracts every close rigidbody to the hit impact, and hurts any element with a health component
- Black hole: the bullet is fired with the usegravity parameter enabled. If a timer does not over before the bullet collides with a surface, the bullet becomes kinematic, and the black hole is enabled with a particles system. It uses an overlap sphere to store all the objects inside a radius to add velocity in the black hole direction. After a while, add to that objects an explosion force and finally the black hole is destroyed. If an object with a health component is inside the radius, is damaged while the black hole stills open.
- Slow down bullet: when it touches an object with the tag slowing, a function is called, to reduce the amount of speed used for that object, so the player can do a determinate action. In this state, the material color of the object are changed to other color, so when the object recovers its speed, after a while, the colors back to their previous value.
- Shut down: allows lock some enemies at the same time, shooting them homing projectiles that destroy them.


Shoot Type (Script)

Script	shootType
Touched	☐
Disable Timer	5
Stop Bullet Timer	1
Layer	Mixed ...
Speed	20
Bullet Mesh	meshBullet
Scorch	scorch
Particles	

- **Touched**, true when the projectile touches a surface.
- **Disable timer**, the timer to destroy the projectile it this doesn't touch any surface.
- **Stop bullet timer**, the time to stop the bullet when the power selected is the black hole.
- **Layer**, the layermask used to calculate the objects affected by the projectile.
- **Speed**, the velocity of the projectile.
- **Bullet mesh**, the mesh of the projectile.
- **Scorch**, the scorch done by the projectile when it is a regular shoot or a grenade and when the projectile hits a surface.
- **Particles**, a list of particles used for every type of projectile.

This projectiles can damage vehicles too, searching for vehicle damage receivers when they hit a surface.

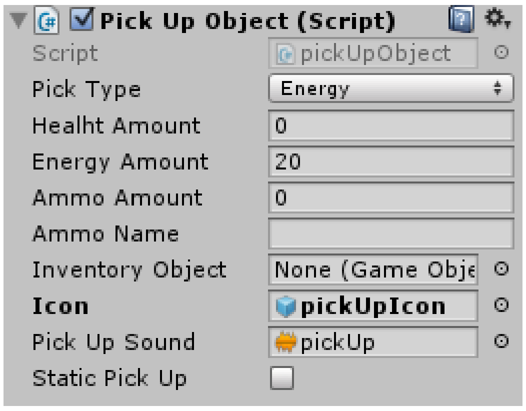
LAUNCHED OBJECTS, it is added to any object launched by the player. Enables the collision between the grabbed objects and the player once that the object collide with any surface. This is done because when the player carries any object, the collision is ignored to avoid that the player be kicked by any of the objects that he carries when he moves or when he throws that objects.

These objects also check if the beaten objects has a health component, to take a certain amount of his life, according to the velocity of the collision.

PICKUP OBJECT, there are pickups, like health and energy. This objects has a trigger, so when the player or the vehicle enters in it, the object moves to the players position, to be grabbed. These objects can be grabbed with the grab objects power.

When the player/vehicle grabs a pickup, this will check if the amount of health, energy or ammo is filled or not, so the player/vehicle only will get the neccessary objects to restore his state. In version 2.3, the player grabbed every pickup close to him. This is also applied when the player is close to more than one pickup, so if the player has 90/100 of health and he is close to two health pickups, he only will grab one of them.

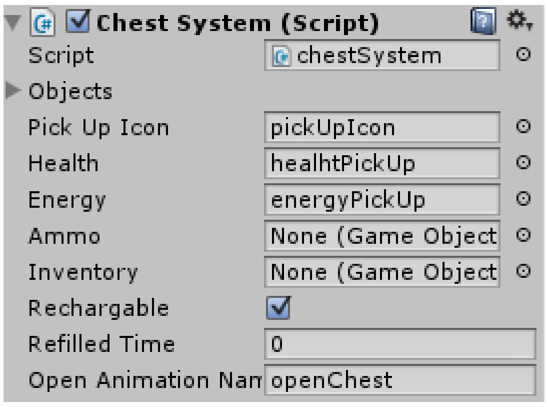
In futures versions, there will be inventory objects and the ammo will can be grabbed by the player to the weapon mode, instead of being only used for vehicles.



Pick Up Object (Script)	
Script	pickUpObject
Pick Type	Energy
Healht Amount	0
Energy Amount	20
Ammo Amount	0
Ammo Name	
Inventory Object	None (Game Obje
Icon	**pickUpIcon**
Pick Up Sound	pickUp
Static Pick Up	☐

- **Pick type**, set which type of pickup is this object: health, energy, ammo or inventory.
- **Health amount**, the amount of health that the pickup restores.
- **Energy amount**, the amount of energy that the pickup restores.
- **Ammo amount**, the amount of ammo for the weapons (not implemented yet)
- **Ammo name**, set the name of the weapon which uses that type of ammo. If the vehicle has a weapon called "Machine Gun", which actually has, the ammo need to has the string "Machine Gun" set, so when the vehicle is inside the trigger, the pickup check if the vehicle has a weapon called like that, so only that kind of ammo is increased. Otherwise, the pickup is not grabbed. In this way you can set as many type of ammo as you want.
- **Inventory object**, (not implemented yet)
- **Icon**, the prefab located in Pick up objects icons, inside hudAndMenus.
- **Pick up sound**, the sound played when the player grabs a pickup object.
- **Static pick up**, If true, the pickup rigidbody is disabled and the trigger radius is set to 1, so to actually use the pick you only can come close to it.

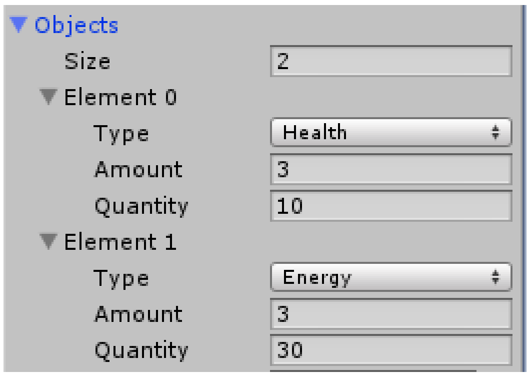
PICKUP CHEST, chest can be created, setting the amount of pickups that has inside it and the value of every pickup, for example 3 health objects which heals the player in 10 units every of them. The chest also can be set to just one use, or if its content is refilled and how much time will take it.



Chest System (Script)	
Script	chestSystem
Objects	
Pick Up Icon	pickUpIcon
Health	healhtPickUp
Energy	energyPickUp
Ammo	None (Game Object
Inventory	None (Game Object
Rechargable	✓
Refilled Time	0
Open Animation Nam	openChest

- **Objects**, the list of pickups that the chest has.
- **Pick up icon**, the icon used to place in the position of the pickup. The gameObject is placed in pick up object icons inside hudAndMenus.
- **Health**, the health pickup prefab.

- **Energy**, the energy pickup prefab.
- **Ammo and inventory** will be add in the version 2.4
- **Rechargeable**, if true, the chest is closed and refilled once the player has opened it.
- **Refilled time**, the time that takes to the chest be refilled. If the value is equal to 0, the chest can be open again immediately.
- **Open animation name**, the name of the animation used to open the chest. The close animation is the open animation reversed.

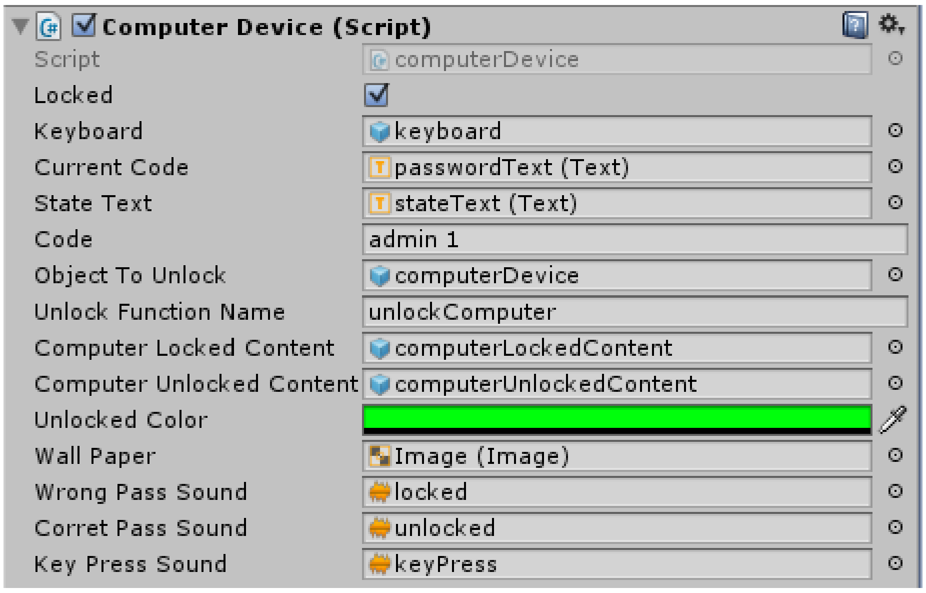


Objects	
Size	2
Element 0	
Type	Health
Amount	3
Quantity	10
Element 1	
Type	Energy
Amount	3
Quantity	30

To add more objects to the chest, set the number pickups types to create in the chest.

Then, for every element, set the pickup type, health, energy, ammo or inventory, the amount of this type of pickup and the value that restore. The limit of pickup objects in the chest is 15, due to the size of the current chest and the pickups. The position of every pickup inside the chest is calculated according to the name of these objects.

COMPUTER DEVICE, the computer has a password, which can be set in the inspector. The computer can have any text as password, number, words or both, and allow to press space, delete or enter. In a smartphone, the keyboard of this terminal can be used by pressing their keys directly with the mouse, or with the finger in a touch device. Once the computer is unlocked, it can show anything you need in screen, like messages, different security cameras, etc…



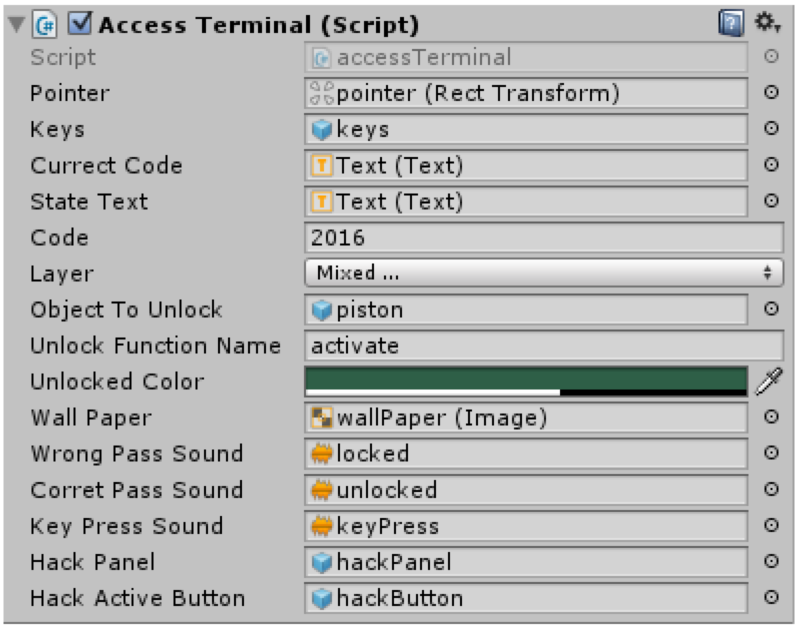
Computer Device (Script)	
Script	computerDevice
Locked	✓
Keyboard	keyboard
Current Code	passwordText (Text)
State Text	stateText (Text)
Code	admin 1
Object To Unlock	computerDevice
Unlock Function Name	unlockComputer
Computer Locked Content	computerLockedContent
Computer Unlocked Content	computerUnlockedContent
Unlocked Color	🟩
Wall Paper	Image (Image)
Wrong Pass Sound	locked
Corret Pass Sound	unlocked
Key Press Sound	keyPress

- **Locked**, if true, the computer is locked initially.
- **Keyboard**, it contains all the keys used to enter the password.
- **Current code**, the text which shows the current code written in the panel.
- **State text**, it shows if the computer is locked or unlocked.
- Code, the current password. It can has any length and it can be text, numbers or both, also using spaces.
- **Object to unlock**¸ the gameObject that is unlocked if the player enters the correct password.
- **Unlock function name**, the name of the function called in the object to unlock.
- **Computer locked content**, the window of the computer showed to unlock the computer.
- **Computer unlocked content**, the window of the computer showed once the computer has been unlocked.
- **Unlocked color**, the color set in some elements of the panel screen when the panel is unlocked.
- **Wall paper**, the wallpaper used in the panel.
- **Wrong pass sound**, the sound played when the password is incorrect.
- **Correct pass sound**, the sound played when the password is correct.
- **Key press sound**, the sound played when a number key is pressed.

ACCESS TERMINAL, like the computer terminal, it uses UI of unity, but it is independent from the main HUD, so also can be modified to make more types of panels and terminals. When the player is close enough, the mouse in the terminal follows the center of the screen (just like in doom 3). Just aim to a number and press the key to activate devices (by default is T, but it can be changed in the key input editor). Also, the panel is already usable with touch controls, by pressing any number with the finger. The password can be set in the inspector and it can has any length.

When the password is correct, it calls to a gameObject function, to activate doors, elevators, etc...

Also, it has an option to hack the terminal, which consists in a secondary panel opened when the player press the hack button icon, which consists in press the correct direction showed in a list of arrows before the time overs. If the arrow are pressed correctly, the pass of the terminal is showed in the hack panel.



Access Terminal (Script)

Script	accessTerminal
Pointer	pointer (Rect Transform)
Keys	keys
Currect Code	Text (Text)
State Text	Text (Text)
Code	2016
Layer	Mixed ...
Object To Unlock	piston
Unlock Function Name	activate
Unlocked Color	
Wall Paper	wallPaper (Image)
Wrong Pass Sound	locked
Corret Pass Sound	unlocked
Key Press Sound	keyPress
Hack Panel	hackPanel
Hack Active Button	hackButton

- **Pointer**, the cursor in the terminal screen, which follows the center of the screen or the finger in a touch device.
- **Keys**, it contains all the key numbers button inside the panel.
- **Current code**, the text which shows the current code written in the panel.
- **State text**, it shows if the panel is locked or unlocked.
- **Code**, the current password. It can has any length.
- **Layer**, the layermask used for the raycast to check the position of the panel where the player is looking at.
- **Object to unlock**¸ the gameObject that is unlocked if the player enters the correct password.
- **Unlock function name**, the name of the function called in the object to unlock.
- **Unlocked color**, the color set in some elements of the panel screen when the panel is unlocked.
- **Wall paper**, the wallpaper used in the panel.
- **Wrong pass sound**, the sound played when the password is incorrect.
- **Correct pass sound**, the sound played when the password is correct.
- **Key press sound**, the sound played when a number key is pressed.
- **Hack panel**, if this access terminal can be hacked, the hack panel is placed here
- **Hack active button**, if this access terminal can be hacked, the hack button in the terminal is placed here.

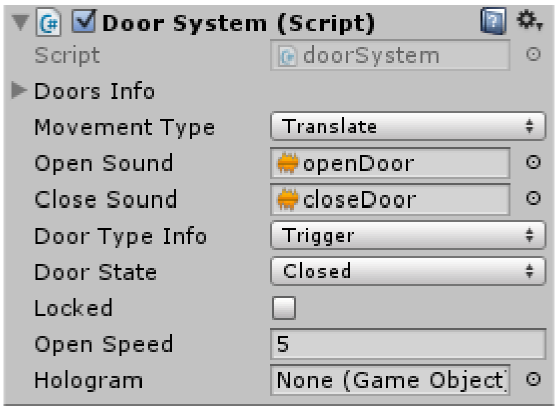
TEXT DEVICES, it allows to show a piece of text in a device, similar to other games, so the player can know more info about any element of the game, tells the main story, etc… It can be used for example for letters. It doesn't need any configuration in the inspector, just change the components inside the gameObject, and set the text that you need.

DOOR SYSTEM, this system allows to make openable doors without animations, just using code. To this, you can create a door with one or more panels, set its pivot position and a final position where the door is translated. Also, you can use an empty gameObject to set a rotation to it, so the door panel with rotate in the same direction. The door can has as many panels as you want, having every one of them a final position or final rotation.

The door can be opened with different ways, using triggers, buttons, shooting to the door or using code panels.

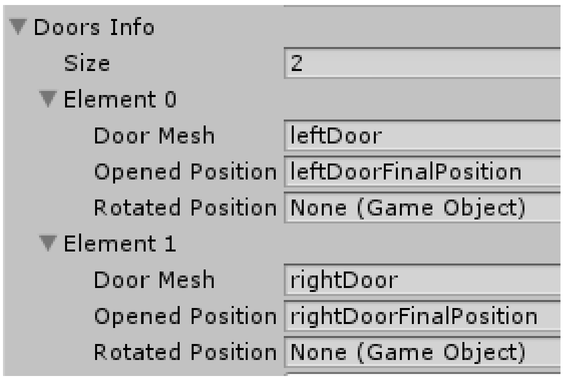
The initial state of the door can be configured, setting opened or closed. Also, the door can be locked, to avoid that the player can open it, before doing a certain action.

In the next update, there will be added more ways to open the door, holograms to show the current state of the door and the option to use animation for the door as well.


Door System (Script)

Script	doorSystem
Doors Info	
Movement Type	Translate
Open Sound	openDoor
Close Sound	closeDoor
Door Type Info	Trigger
Door State	Closed
Locked	☐
Open Speed	5
Hologram	None (Game Object)

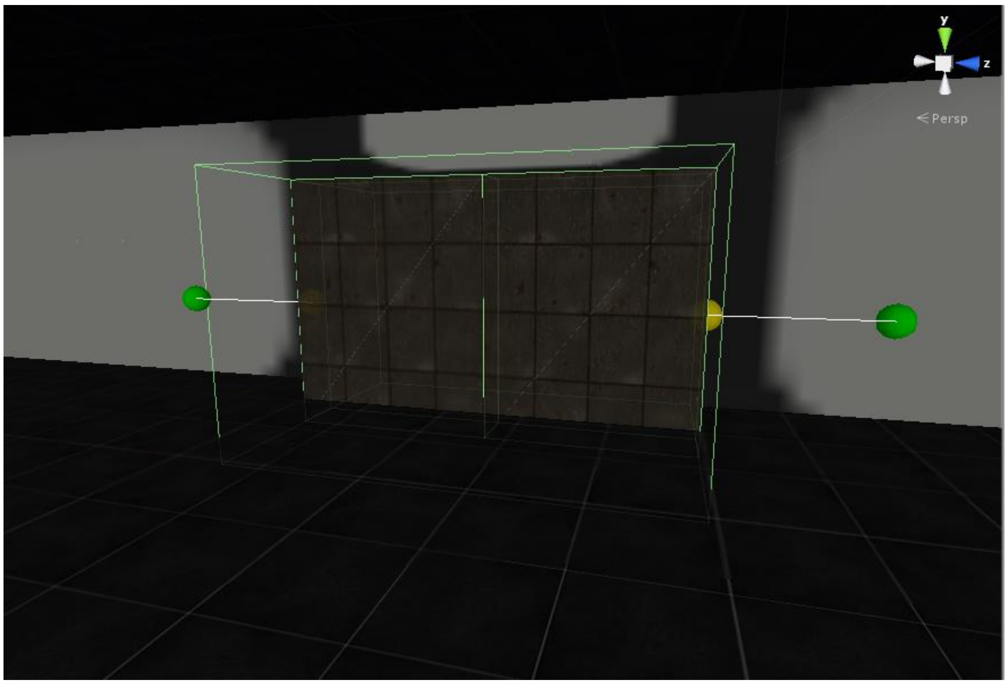
- **Doors info**, the list of panels that the door has, with the final positions or rotation of every panel.
- **Movement type**, the door is translated or rotated.
- **Open and close sound**, the sound effect played when the player enters or exits the door.
- **Door type info**, how the door is opened, using triggers, buttons, shoots or hologram.
- **Locked**, set if the initial state of the door is locked, to open it with an access terminal, you have to set the door in the accessTerminal inspector, in the field objectToUnlock and call the function unlockDoor in the accessTerminal inspector.
- **Open speed**, the velocity to open and close the door.
- **Hologram**, if the door type is hologram, here is placed the hologram object.


Doors Info

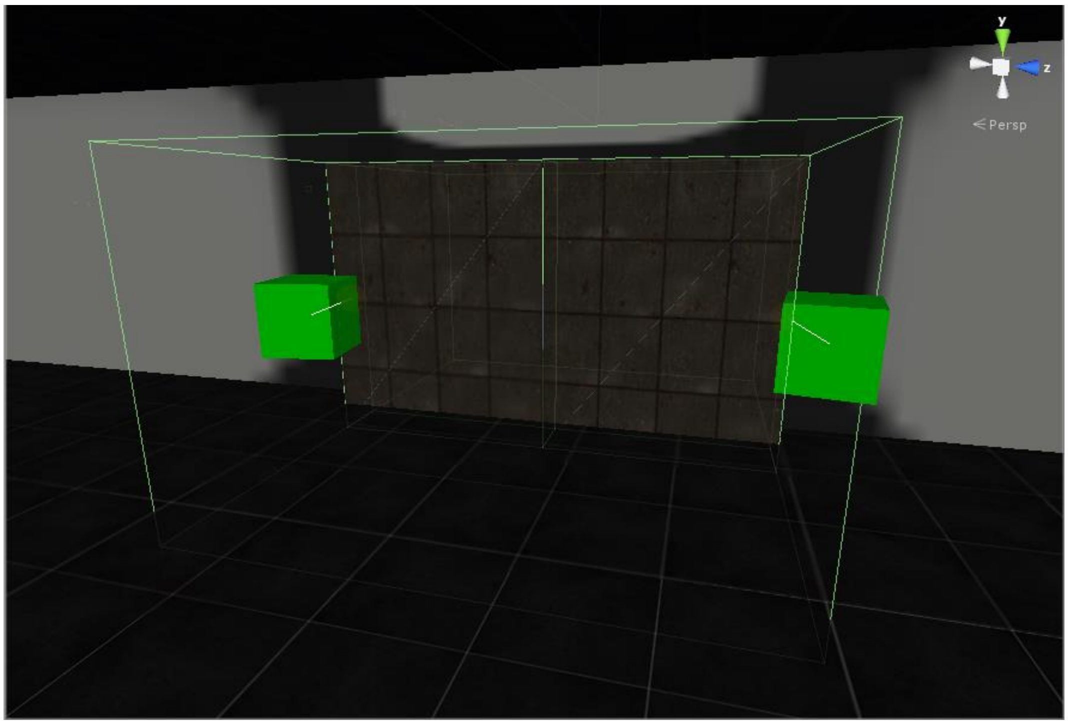
Size	2
Element 0	
Door Mesh	leftDoor
Opened Position	leftDoorFinalPosition
Rotated Position	None (Game Object)
Element 1	
Door Mesh	rightDoor
Opened Position	rightDoorFinalPosition
Rotated Position	None (Game Object)

For every panel in the door, you have to define the door mesh, which is the empty element that acts as the pivot door and which contains the panel mesh inside of it and the final position of the panel.

In this capture you can see a door with two panels, which opens by translating every panel pivot to the final position in every side. The script has a gizmo to show in editor mode the pivot position of every panel and the final position where they are translated, by placing spheres in those position and a line joining every pivot and final position.



In this other capture, you can see a rotating door. The boxes show the final rotation that the door panels will have.



SETGRAVITY AND SETGRAVITTONORMAL, both uses triggers to set the gravity of the player. The first one is an arrow that makes the player moves in the direction where the arrows aims. The second makes the gravity of the player back to normal.

Also, when the player is driving, if the vehicle touches these triggers, their gravity will be modified too.

ARTIFICIAL OBJECT GRAVITY, when the player is in aim mode with a grabbed object, if he press the left mouse of the button, the object is dropped, disabling its regular gravity, adding force in the direction where the camera was looking, and when the objects touches a surface, the normal of that surface it Is its new gravity.

The script uses a raycast to check if the object is in its new ground, else a force is added in its new gravity direction.

The gravity of the turrets can also be changed, like any other object. When the turret hits a surface, the turret is rotated to that surface, to allow the player to set it in any place.

To set the gravity object to its regular gravity, grab it again and set its direction to the ground, so when the object collides with the ground, the gravity is enabled again and the script is removed.

This system allows the player to create different puzzles, with buttons located in the walls for example.

This system is used too by the barrel launched by the car weapon, when the vehicle has a different gravity, so the barrel gravity is changed according to the gravity of the vehicle. In this way, the barrel can make a parable in any surface and direction.

CHECK COLLISION TYPE, it allows to check different types of collision, with colliders and triggers, on exit and on enter, calling a function according to how it is configured. Every type of collision call its own function, which is a string field in the editor. Also, can be check if the object collides with a specific object. This script is used in different situations, like joining objects.

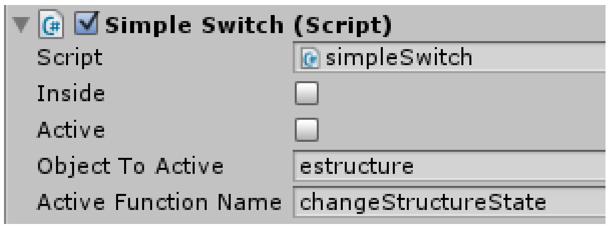
RAIL MECHANISM, it is used when the player grabs a rail object, with the name mechanism. It checks if the objects is too far from its original position, to reset its position. If the object reaches its final position, the script is disabled, because the object has been engaged. Also has a rotor gameObject, which is used to connect the mechanism and it can be slowed down, if the object is different to null. Else the rail only have to be moved to its final position.

MECHANISM PART, it is the script called when the mechanism is engaged, connecting all the system.

PUT GEAR, it sets the gear in its place, when the gear touches the trigger, inside the rail object in the scene.

SIMPLE SWITCH, when the player is inside the trigger of an object with the device tag, like the hackable device in the back of every turret, the player can press T to activate that device. To work, the trigger of the object with this script also activates the using Devices System script in the player that set the icon of a usable device in the script, and call the function to activate the device.

Finally, a bool parameter in this script is checked by another script, so when the bool parameter is true, another objects is connected.

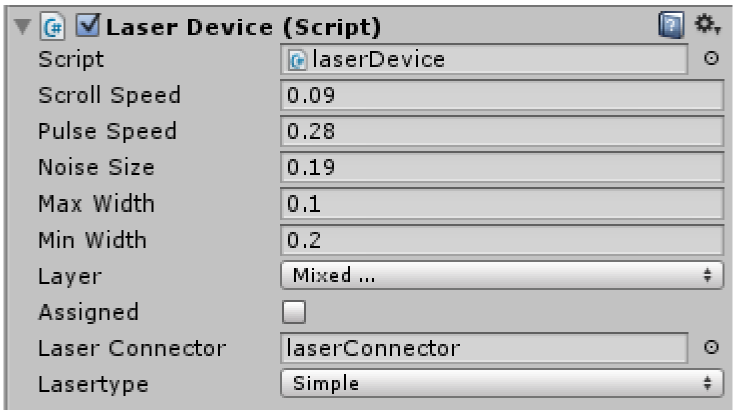


Simple Switch (Script)	
Script	simpleSwitch
Inside	☐
Active	☐
Object To Active	estructure
Active Function Name	changeStructureState

- **Inside and active** are just to see the current state of the switch, if the player is inside the trigger or not and if the button has been pressed.
- **Object to active**, the gameObject that the switch enables or disables.
- **Active function name**, the function called in the object to active.

SLOW OBJECTS COLOR, when the player uses its power to slow down an object or an enemy, this script is attached in that object, to change all the materials color inside the object. When the effect is over, the materials color are set to its regular color. This is just to the player can see what object is slowed down, and for how long.

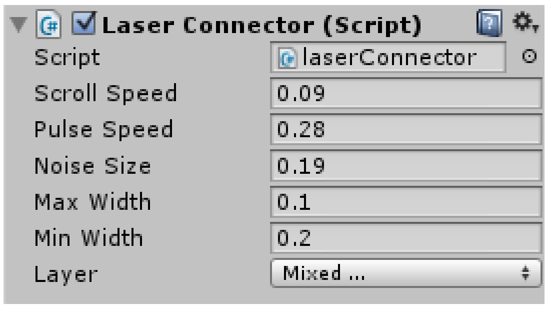
LASER DEVICE, is the laser that the player can use to activate other devices, use the cube refraction, or use it as a weapon. It checks with a raycast if the player touches the laser, calling the functions to activate the laser in the hand of the player. Now, it allows to set if the laser that the player uses, can be refracted or not.



Laser Device (Script)	
Script	laserDevice
Scroll Speed	0.09
Pulse Speed	0.28
Noise Size	0.19
Max Width	0.1
Min Width	0.2
Layer	Mixed ...
Assigned	☐
Laser Connector	laserConnector
Lasertype	Simple

- **Scroll and pulse speed**, noise size and max and min width are used to animate the laser.
- **Layer**, is the layermask used to detect the player and to check if the laser collides with a collider.
- **Assigned**, if true the laser has been connected to a laser receiver or a refraction cube.
- **Laser connector**, the laser created when the player has connected the laser device to a refraction cube or a laser receiver.
- **Lasert type**, the laser in the hand of the player will have or not refractions.
-

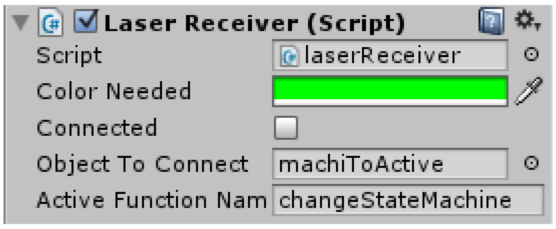
LASER CONNECTOR, when the laser of the player is used to connect other device, or touches a cube refraction, the laser deflected is the laserconnector, pausing the laser device, and checking if any object cross the laser, to deactivate it. The laser connector has the same parameters than laser device.



Laser Connector (Script)

Script	laserConnector
Scroll Speed	0.09
Pulse Speed	0.28
Noise Size	0.19
Max Width	0.1
Min Width	0.2
Layer	Mixed ...

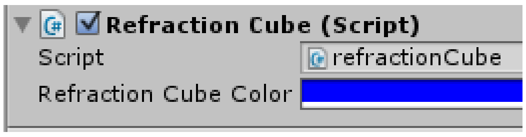
LASER RECEIVER, when the player's laser detects a laser receiver, it checks the color needed to connect the device, if the color of the receiver is equal to the color of the current laser of the player, the laser is reflected. Then if an object to connect and an active function name has been placed in the inspector, the laser receiver call this function in the object to connect, so you can use this system to connect a door, a device, etc..... If the laser connected to laser receiver is disabled, the function of the object to connect is called again, to disable that object too.



Laser Receiver (Script)

Script	laserReceiver
Color Needed	🟩
Connected	☐
Object To Connect	machiToActive
Active Function Nam	changeStateMachine

REFRACTION CUBE, it is a cube to reflect and change the color of the laser, according to the color configured in the cube inspector.

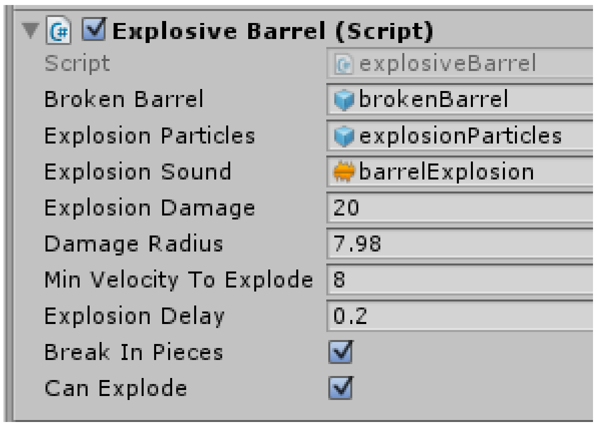


Refraction Cube (Script)

Script	refractionCube
Refraction Cube Color	🟦

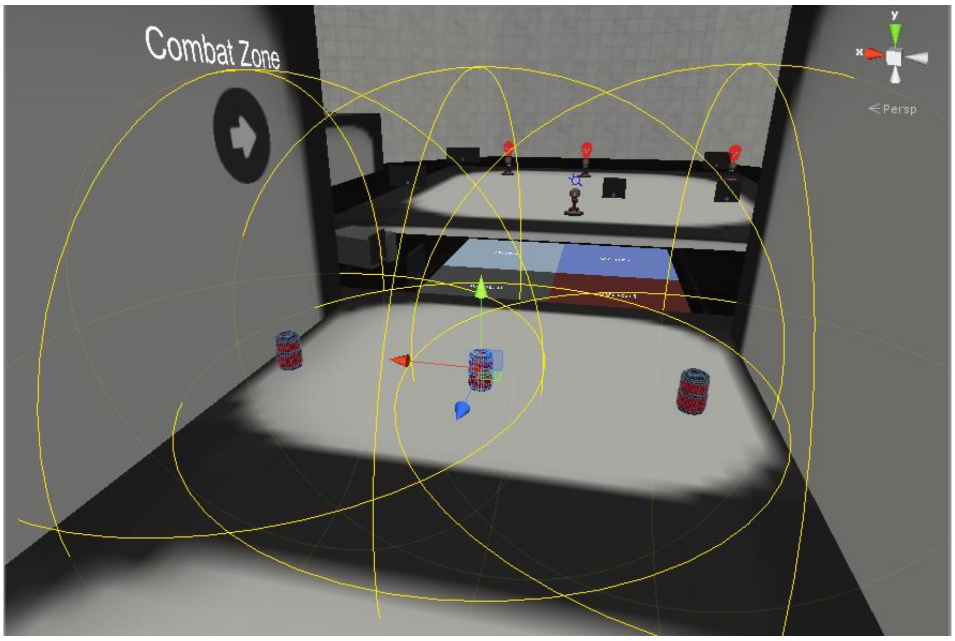
EXPLOSIVEBARREL, These barrels can be grabbed, dropped and thrown by the player. When they collide with a surface or an object, they explode, breaking the mesh of the barrel in pieces which disappear in some seconds, and also, it can be added a drop pickups option, so now there are more ways to get these pickups.

The barrels damages and pushes any object close to its explosion, which radius can be configured and can be destroyed just by shooting, throwing any object or hitting with a vehicle. Chain reactions can be done with them.


Explosive Barrel (Script)	
Script	explosiveBarrel
Broken Barrel	brokenBarrel
Explosion Particles	explosionParticles
Explosion Sound	barrelExplosion
Explosion Damage	20
Damage Radius	7.98
Min Velocity To Explode	8
Explosion Delay	0.2
Break In Pieces	✓
Can Explode	✓

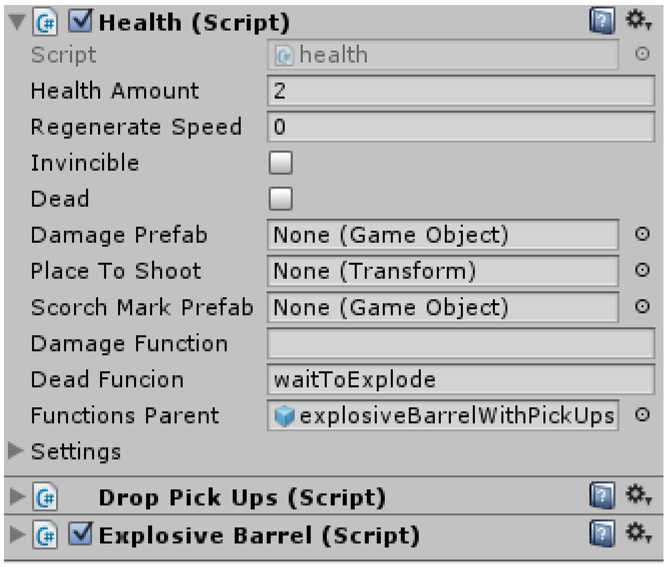
- **Broken barrel**, the prefab model broken in pieces created when the barrel explodes.
- **Explosion particles**, the prefab particles created when the barrel explodes.
- **Explosion sound**, the sound played at explosion.
- **Explosion damage**, the damage applied to every object with a health or vehicle health component.
- **Damage radius**, field of action to search objects which can be damage.
- **Min velocity to explode**, the minimum velocity at collision enter necessary to explode.
- **Explosion delay**, if the barrel is damaged, the amount of time until the barrel explodes when its health component reaches 0.
- **Break in pieces**, created or not the broken barrel model when the barrel explode.
- **Can explode**, used to avoid the barrel explode at collision when the player carries it.

In this picture you can see the damage radius of three barrels, using a gizmo line sphere to draw it. In this way, when a barrel explodes, it searches objects inside that radius to damage it. In this case, if one barrel explodes, a chain reaction happens, exploding the other two barrels. The explosion delay allows to make it more real the reaction chains, because if the delay would 0, the explosion after the first barrel would be immediate, so it is better to use a delay value.



This is the inspector of a barrel. The barrel has a health component with a low health amount, so it will explode with just one shoot from player for example. When the health reaches 0, the function waitToExplode is called, waiting the explosion delay to be exploded.

Also, it can be attached a drop pickups component, so like turrets, when the barrel explode it drops different pickups configured in the inspector.



Health (Script)

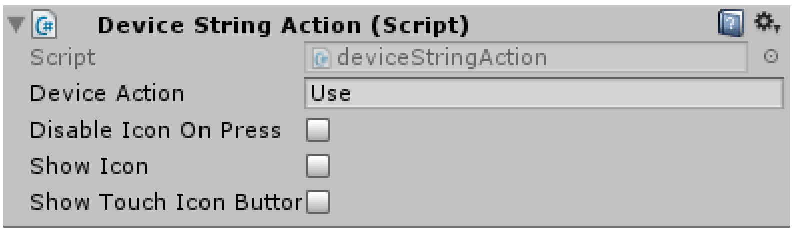
Script	health
Health Amount	2
Regenerate Speed	0
Invincible	☐
Dead	☐
Damage Prefab	None (Game Object)
Place To Shoot	None (Transform)
Scorch Mark Prefab	None (Game Object)
Damage Function	
Dead Funcion	waitToExplode
Functions Parent	explosiveBarrelWithPickUps
Settings	

Drop Pick Ups (Script)

Explosive Barrel (Script)

DEVICE STRING ACTION, this script has to be in every object used for the player as a device, with the tag device and in vehicles, so the player can get in and off from them. It is used to set if the icon of the screen is enabled to show the player that an object can be used, setting the text that the icon shows, for example, activate platform, open door, etc…

Also, the icon is disabled or not when the player press the interaction button. Finally, it enables to show the touch icon button to uses a device.

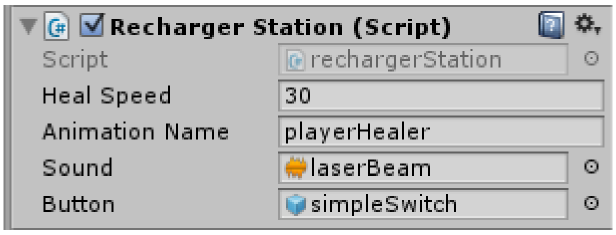


Device String Action (Script)

Script	deviceStringAction
Device Action	Use
Disable Icon On Press	☐
Show Icon	☐
Show Touch Icon Buttor	☐

RECHARGER STATION, like in other games, if the player has not completed his health, energy or both, he can enter in a recharger station and press the button to refill them. If both bars are filled, the station cannot be used.

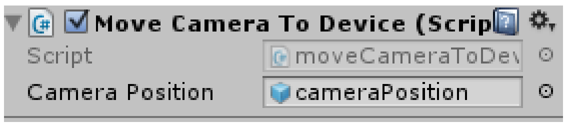
Once the bars are filled, the station stops the recharging.


Script	rechargerStation
Heal Speed	30
Animation Name	playerHealer
Sound	laserBeam
Button	simpleSwitch

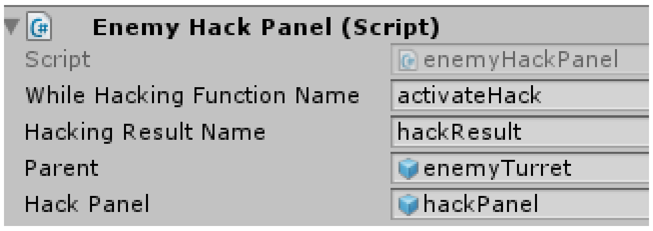
- **Heal speed**, the speed which the health and energy are refilled.
- **Animation name**, the animation made by the station when is being used.
- **Sound**, the sound played while the player is being recharged.
- **Button**, the simple switch object use to activate the station.

MOVE CAMERA TO DEVICE, this script is attached to those devices that need to be placed in front of the camera and the mouse of touches are necessary to use it, like computers, hack terminals, etc… Just attach the script and set the camera position transform where the main camera will be placed. When the camera is displaced to the device, the player camera and the player controller is disabled along with any other action of the player, until he disable the device.

Then, the camera is placed again in the player camera gameObject and every component is enabled again.


Script	moveCameraToDe\
Camera Position	cameraPosition

ENEMY HACK PANEL, used to hack enemies. When the hack start, the enemy behavior is paused, until the hack is done or fail. This script it pauses the enemy and send the hack result to it.


Script	enemyHackPanel
While Hacking Function Name	activateHack
Hacking Result Name	hackResult
Parent	enemyTurret
Hack Panel	hackPanel

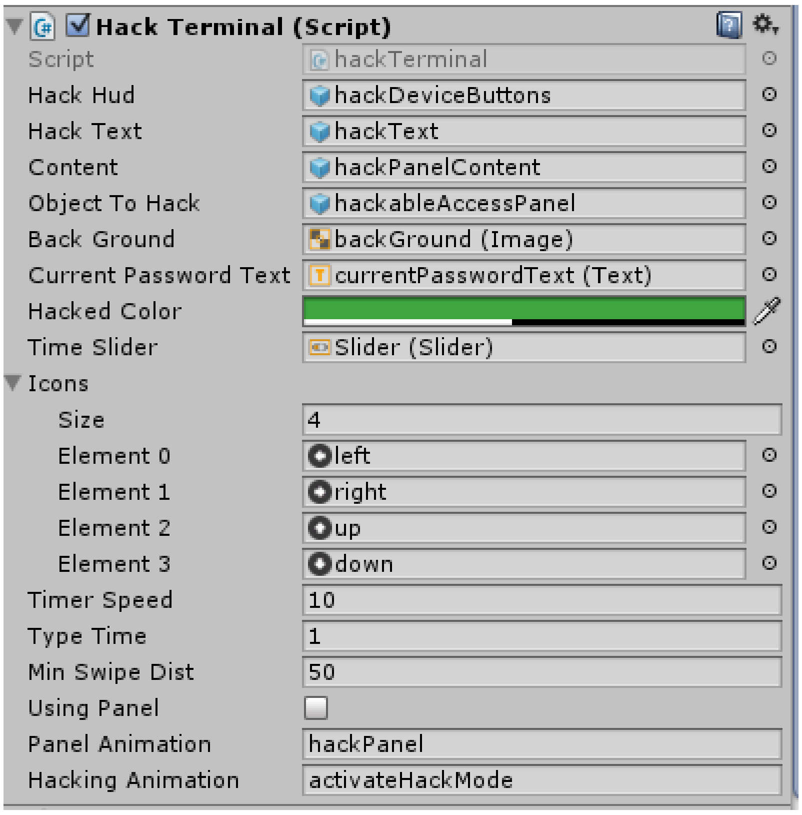
- **While hacking function name**, the name to active the hack terminal.
- **Haching result name**, the function sent to the enemy with a Boolean true or false.
- **Parent**, the enemy to send the message.
- **Hack panel**, the hack panel with the hack terminal component.

HACK TERMINAL, when the player reaches the back part of an enemy turret or is going to hack a device and press the use device key, a panel in that enemy or device is showed, with a time slider and four arrows, with a random direction, different and random in every hacking. To hack it, the player has to press WASD in the direction of the current arrow, before the slider is full.

If the player press a wrong direction, the hacking is over, and the turret will detect and shoot the player, or the device will expulse it, starting again. If the player hack the turret correctly, it becomes his ally. In this case, the name of the turret, its health slider and its radar icon will be changed also. If the object to hack is a code terminal device, it will show the current password of it.

The directions also can be done by swiping, if the game is played in a touch device.

Finally, when the player is close to a hackable device, an icon is showed in the canvas component, following the position of that device, so the player can see it.



Hack Terminal (Script)

Script	hackTerminal
Hack Hud	hackDeviceButtons
Hack Text	hackText
Content	hackPanelContent
Object To Hack	hackableAccessPanel
Back Ground	backGround (Image)
Current Password Text	currentPasswordText (Text)
Hacked Color	🟩
Time Slider	Slider (Slider)
Icons	
Size	4
Element 0	left
Element 1	right
Element 2	up
Element 3	down
Timer Speed	10
Type Time	1
Min Swipe Dist	50
Using Panel	☐
Panel Animation	hackPanel
Hacking Animation	activateHackMode

- **Hack hud**, the HUD that contain the elements of the hack system enabled when the player is hacking an enemy.
- **Hack text**, the text that show if the hacking has been done correctly or not.
- **Content**, the full content of the hack terminal, used it to enable or disable it.
- **Object to hack**, set here the object that will receive the hack result from the terminal.
- **Background**, the red panel in the background of the terminal, which color will be changed when the hack is done.
- **Current password text**, the text object to show the password of the access terminal.
- **Hacked color**, the color set in the background when the hack is done.
- **Time slider**, the slider that show the remaining time to hack the turret.
- **Icons**, the textures used for the hacking minigame, to show the direction of the button to press.
- **Timer speed**, the speed that the slider increases its value.
- **Type time**, the time used to set the hacking result in the screen.
- **Min swipe dist**, the minimum distance to make the swipe while the player is hacking in a touch device.
- **Using panel**, the hack panel is being used or not.

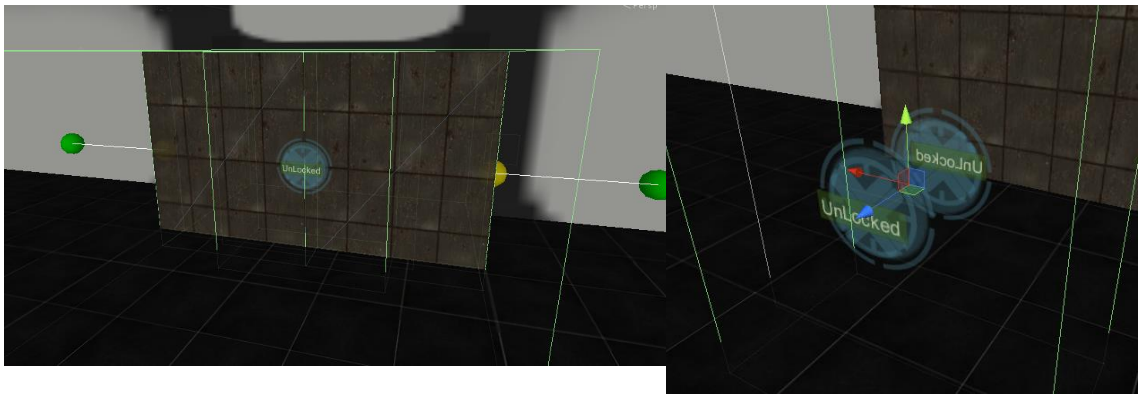
- **Panel animation**, the name of the animation played to show the hack terminal, which is inside other object.
- **Hacking animation**, the animation made by the canvas component when the hack starts or stops.

HOLOGRAM DOOR, it works like in dead space, when you are inside the trigger of the hologram, press the use device button (T by default), and the door is opened. When the player is inside the trigger of the hologram, the rotating animation is stopped and the central ring animation is played, changing the text of the hologram.

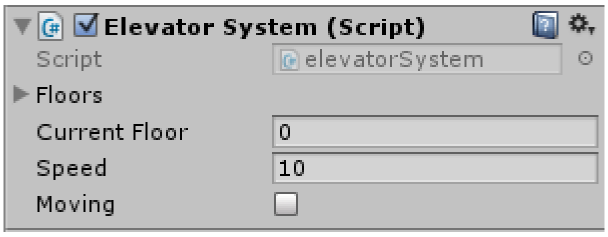
When the player exits from the trigger, the hologram rotating animation is played again, changing the text again for the state of the door, locked or unlocked, according to the configuration of the inspector.

If he door is opened, the hologram fades, until the player exits from the doors trigger, and once is closed, the hologram is visible again.

In these pictures, you can see how the hologram is placed in a door and the two sides of the hologram, so it is visible from the both sides of the door. The hologram has the tag device.

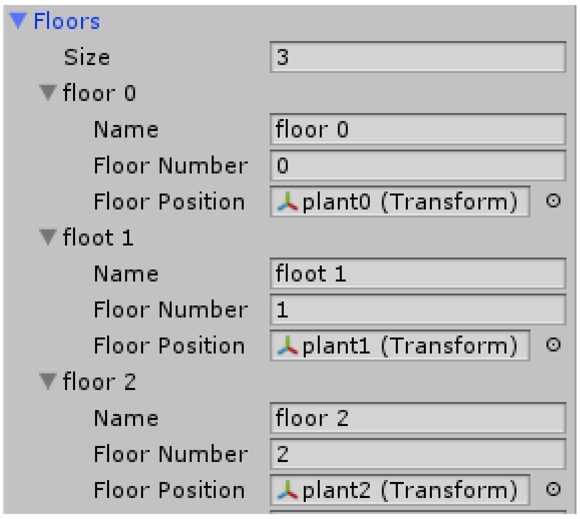


EVELATOR SYSTEM, it allows to set any amounts of floors to move as waypoints, and can move in any direction making that the same elevator can move in horizontal and vertical direction if it is configured like that. To simple switch buttons are placed inside the elevator, one to move to the next floor and another to move to the previous one.


Elevator System (Script)

Script	elevatorSystem
Floors	
Current Floor	0
Speed	10
Moving	☐

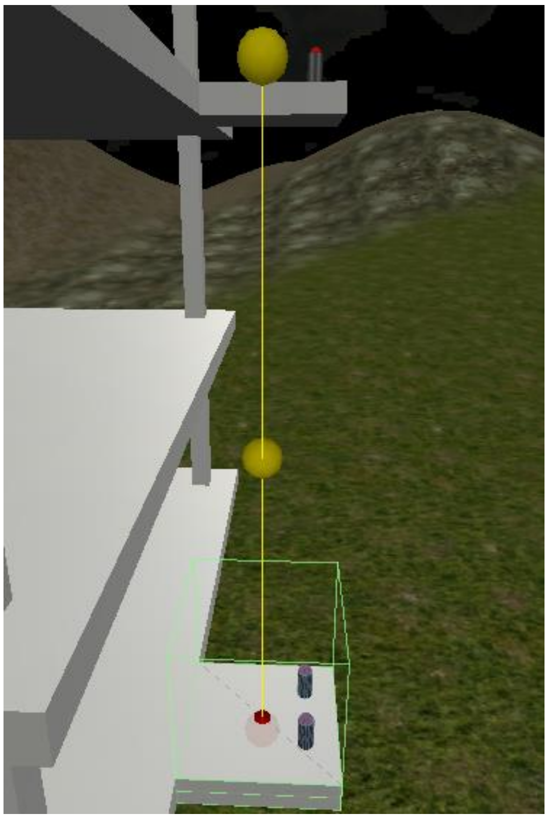
- **Current floor**, the current floor where the elevator is.
- **Speed**, the movement speed of the elevator.
- **Moving**, the elevator is moving or not.

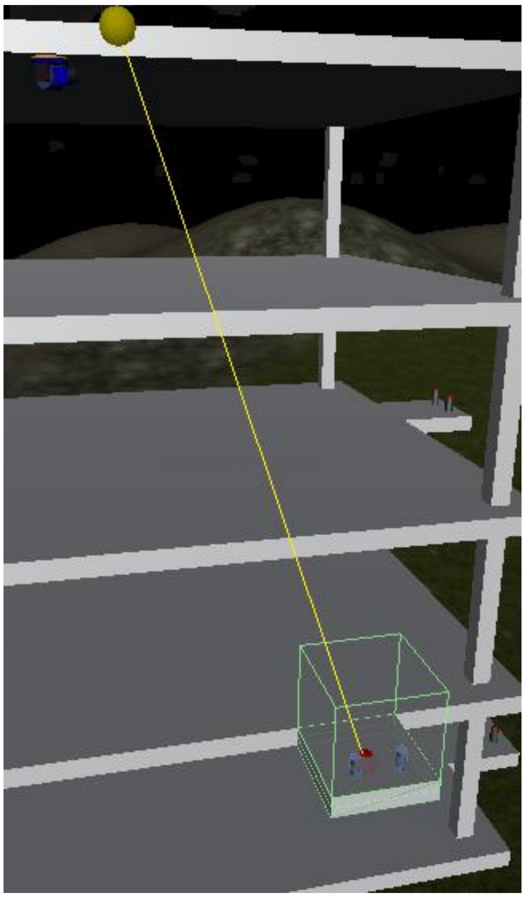


```
▼ Floors
    Size              3
  ▼ floor 0
      Name            floor 0
      Floor Number    0
      Floor Position  plant0 (Transform)
  ▼ floot 1
      Name            floot 1
      Floor Number    1
      Floor Position  plant1 (Transform)
  ▼ floor 2
      Name            floor 2
      Floor Number    2
      Floor Position  plant2 (Transform)
```

Now, you only has to set the number of floors, and inside of everyone set the name of the floor, the floor number and the position where the elevator will be translated.

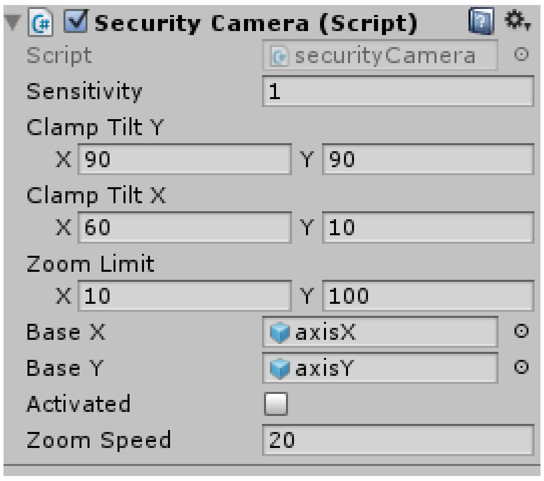
In the left picture is showed the gizmo of the elevator. Every yellow sphere shows every floor position and the line joins every floor position with the next. The current floor is showed as a red sphere. In the right, you can see that a floor can be placed in any position, making any type of movement.





SECURITY CAMERA and **SECURITY CAMERA CONTROL**, this camera is only moved by the security camera control component placed in one computer device, which can has one or more cameras to switch, and the control is a joystick attached in the keyboard which rotates along with the mouse or the finger, rotating the security camera too. Also, it has two buttons to zoom in and out. The camera component inside the model is only enabled when the controls are being used, setting the render texture in a plane inside the computer.

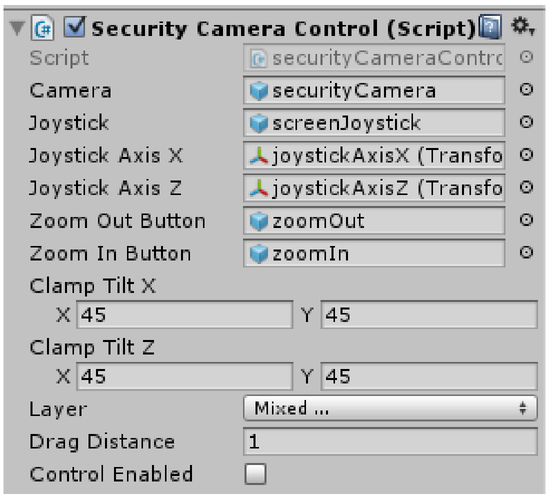
Both script are used together, for every camera used in a computer the security camera script in the camera model and the security camera control in a gameObject inside the computer.



Security Camera (Script)

Script	securityCamera
Sensitivity	1
Clamp Tilt Y	X 90 Y 90
Clamp Tilt X	X 60 Y 10
Zoom Limit	X 10 Y 100
Base X	axisX
Base Y	axisY
Activated	☐
Zoom Speed	20

- **Sensitivity**, a multiplier value to apply rotation to the camera.
- **Clamp tilt X and Y**, the rotation limit for the x and y local axis.
- **Zoom limit**, the minimum and maximum values of the field of view in the camera component, used for the zoom.
- **Base X and Y**, the transform parts of the camera to rotate the x and the y axis.
- **Activated**, the camera is being used
- **Zoom speed**, a speed multiplier for the zoom state.



Security Camera Control (Script)

Script	securityCameraContrc
Camera	securityCamera
Joystick	screenJoystick
Joystick Axis X	joystickAxisX (Transfo
Joystick Axis Z	joystickAxisZ (Transfo
Zoom Out Button	zoomOut
Zoom In Button	zoomIn
Clamp Tilt X	X 45 Y 45
Clamp Tilt Z	X 45 Y 45
Layer	Mixed ...
Drag Distance	1
Control Enabled	☐

- **Camera**, the security camera model which has the security camera component.

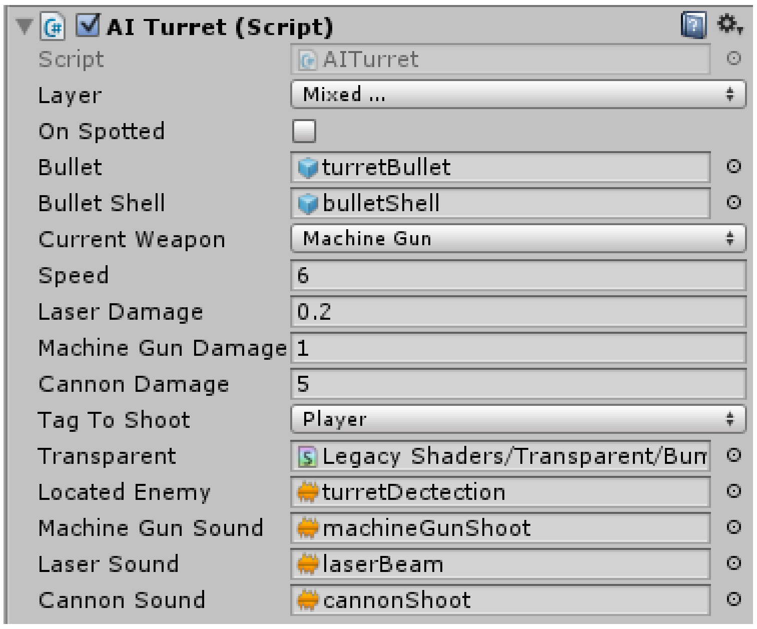
- **Joystick**, the joystick in the keyboard or the computer which has the trigger to detect if the mouse or the finger has pressed it.
- **Joystick axis Z and X**, the transform gameObjects of the joystick to rotate it according to the mouse/finger position.
- **Zoon out and in button**, the buttons in the keyboard to use the zoom, red to zoom out, green for zoom in.
- **Clamp tilt X and Z**, the max amount of rotation added to the joystick.
- **Layer**, the layer to check if the mouse/finger has pressed above the joystick trigger.
- **Drag distance**, the max amount of drag in the joystick.
- **Control enabled**, the control is being used.

TURRETS

AI TURRET, the turret has an advanced AI and some features, which are the next:

- Three different weapons: laser, machine gun and cannon. The weapon used in every turret can be changed in the editor and in run time, disabling the current weapon and activating the new chosen one. In the inspector, it can be set the damage of every weapon. Every weapon has a shoot rate: the laser is activated continuously, the cannon takes 1 second between fires and the machine gun takes 0.1 seconds. In this time, the turret uses a raycast, so when the turret detects the enemy during that time, it will fired.
- Enemy or ally: every turret can be an enemy which will shoot the player and any other object with the tag friend. In the other hand, the turret can be a friend, which will shoot to any object with the enemy tag. In this state, the turret can be grabbed by the player, in the regular mode, carrying the object in the sides of the player, or in the aim mode. In this mode, the gravity of the turret can be changed, like any other object, by pressing the left mouse button. When the turret touches a surface, the turret rotates to the normal direction of that surface. The names and slider color of the turret are different, so the player can distinguish every type of turret.
- List of detected enemies: the turret has a sphere collider trigger inside of it, checking if an object with the current tag to shoot has entered. Every enemy inside the trigger is stored in a list, checking which enemy is the closest, to shoot him. If the enemy is destroyed of leave the field of view of the turret, it is removed from the list, so the turret will aim to the next enemy closest to it.
  If the list of enemies is empty, the turret rotates to search new targets, using a capsule collider, so if object with the current enemy tag triggers it, the turret will check that object, what takes 2 seconds. If the object does not leave, it is stored in the list enemies, so the turret will shoot it. Else the turret continues rotating, ignoring the object.
- Search an enemy who damaged the turret: when the turret is damaged by an object with an enemy tag, the field of view of the turret is set to the distance between the object and the turret, so it will fired that object, storing it in the list of enemies. If the turret is damaged by a not enemy tag object, the turret will not search anything. For example, if the player fires to an ally turret, the turret will not damage him, because the friendly fire is allowed.
- Hackable turret: if the turret has the tag enemy, the turret can be hacked by the player. To do this, the player can crouch and go to the back of the turret, where the device to hack is located. The other way is by using the slow down power, and quickly, go the

same place. Then press T, and press the WASD in the same order that the arrows indicate. If the player get up, or he press the wrong key, the turret will attack him. Else the turret becomes an ally.

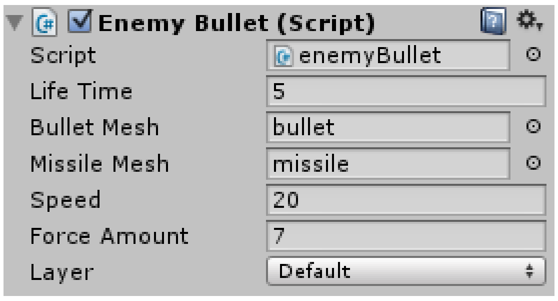

AI Turret (Script)

Script	AITurret
Layer	Mixed ...
On Spotted	☐
Bullet	turretBullet
Bullet Shell	bulletShell
Current Weapon	Machine Gun
Speed	6
Laser Damage	0.2
Machine Gun Damage	1
Cannon Damage	5
Tag To Shoot	Player
Transparent	Legacy Shaders/Transparent/Bum
Located Enemy	turretDectection
Machine Gun Sound	machineGunShoot
Laser Sound	laserBeam
Cannon Sound	cannonShoot

- **Layer**, the layer used to check if the turret has seen to the player, other enemy turret, or hit a surface.
- **On spotted**, the turret has located one or more enemies inside its field of view and it will attack to the closest enemy.
- **Bullet**, the prefab projectile fired by the bullet.
- **Bullet shell**, the shell for every bullet fired.
- **Current weapon**, an enum to select the main weapon used by the turret, machine gun, cannon or laser.
- **Speed**, the rotation speed of the turret.
- **Laser damage**, the damage produced by the laser,
- **Machine gun damage**, the damage produced by the machine gun.
- **Cannon damage**, the damage produced by the cannon.
- **Tag to shoot**, an enum to set if at the start of the game, the turret has to shoot to any object with the tag Player, so the turret is an enemy and it will shoot to the player, or the turret has to shoot to any object with the tag enemy, so the turret is an ally and it will shoot to enemies.
- **Transparent**, the shader applied to every turret mesh when the turret is destroyed, to fade all its elements.
- **Located enemy, machine gun, cannon and laser**, sounds used for every weapon in the turret.

ENEMY BULLET, when a turret fires, the projectile can be a bullet or a missile, and according to the type, its renderer is enabled. The bullet moves in straight line and the missile follows its target. If the projectile does not collides in a certain amount of time, the projectile is destroyed. The damage of the bullets and missiles can be set in the inspector.

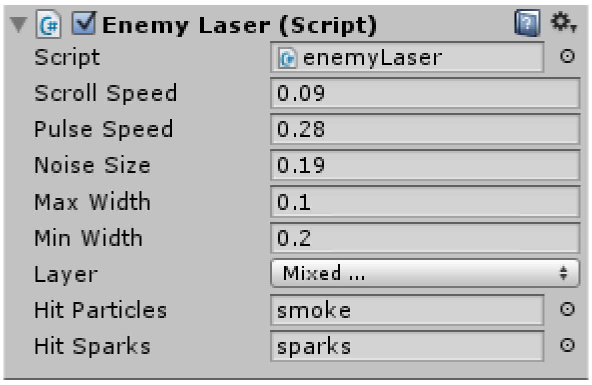
The bullet only damages the target while the missile damages all the objects with health inside a radius. If the enemy projectile touches the shield of the player, the projectile is grabbed by the shield, so the player can fire all the projectiles in his field, back them to the objects that fired them. If the object what fired the projectile has been destroyed, the projectiles are fired in the camera direction. In this case, the missiles use an overlap sphere to check if an enemy is close to it, so the enemy found becomes its target.


Enemy Bullet (Script)

Script	enemyBullet
Life Time	5
Bullet Mesh	bullet
Missile Mesh	missile
Speed	20
Force Amount	7
Layer	Default

- **Life time**, if the bullet doesn't hit any surface and it has passed five seconds since it has been fired, the bullet is destroyed.
- **Bullet mesh**, the bullet mesh inside the bullet prefab.
- **Missile mesh**, the missile mesh inside the bullet prefab.
- **Speed**, the movement velocity of the bullet.
- **Force amount**, the force applied to a rigidbody touched by the bullet.
- **Layer**, the layermask used for the sphere cast used to detect closed targets.

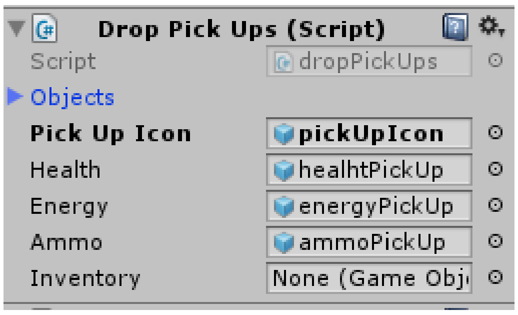
ENEMY LASER, it works like the laser of the enemy, but it only searches the objects with the current enemy tag of the turret, which can be Player and friend or enemy. The enemy laser has the same parameters that the rest of the lasers, but adding a hit particles and sparks, the same type of particles used in the player's laser.


Enemy Laser (Script)

Script	enemyLaser
Scroll Speed	0.09
Pulse Speed	0.28
Noise Size	0.19
Max Width	0.1
Min Width	0.2
Layer	Mixed ...
Hit Particles	smoke
Hit Sparks	sparks

TURRET FOV, attached in a sphere collider trigger, calling a function in the AITurret script every time an object enters or exits of it. The AITurret script will check if that object is an enemy.

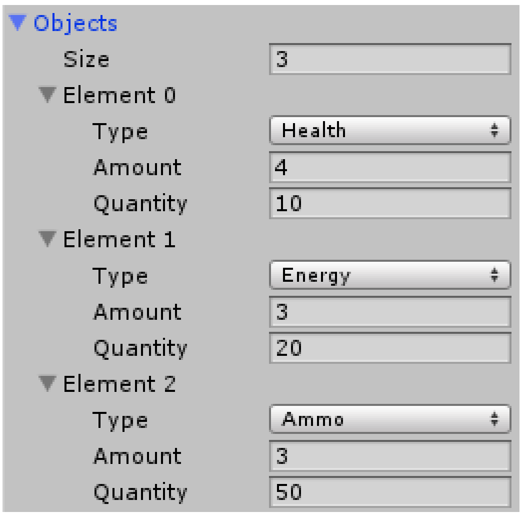
DRO PPICKUPS, this script can be used when a turret explodes, sending a message to call the drop pickups function. When this happens, the script instantiated in the turret position every pickup configured in the inspector, setting their recharge values. It can add as many objects as needed, with health, energy or ammo (inventory type will be added in version 2.4).



Drop Pick Ups (Script)

Script	dropPickUps
Objects	
Pick Up Icon	**pickUpIcon**
Health	healhtPickUp
Energy	energyPickUp
Ammo	ammoPickUp
Inventory	None (Game Obj

The gameObjects placed in the inspector are the pickups prefabs, located in the prefabs folder.



Objects	
Size	3
Element 0	
Type	Health
Amount	4
Quantity	10
Element 1	
Type	Energy
Amount	3
Quantity	20
Element 2	
Type	Ammo
Amount	3
Quantity	50

The steps to add objects is like in the chests system. To add objects to the drop list, set the number pickups types to create in the chest.

Then, for every element, set the pickup type, health, energy, ammo or inventory, the amount of this type of pickup and the value that restore. The limit of pickup objects in the chest is 15, due to the size of the current chest and the pickups. The position of every pickup inside the chest is calculated according to the name of these objects.

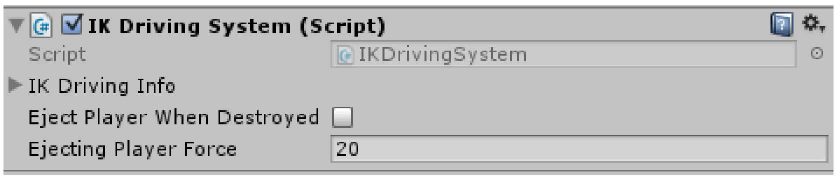
This system is used too for explosive barrel, so when they explode they can drop pickups, or just explode.

VEHICLES

IK DRIVING SYSTEM, this script is used to configure the different player's limbs positions when he is driving, allowing to configure any position and rotation of the player according to the type of vehicle.

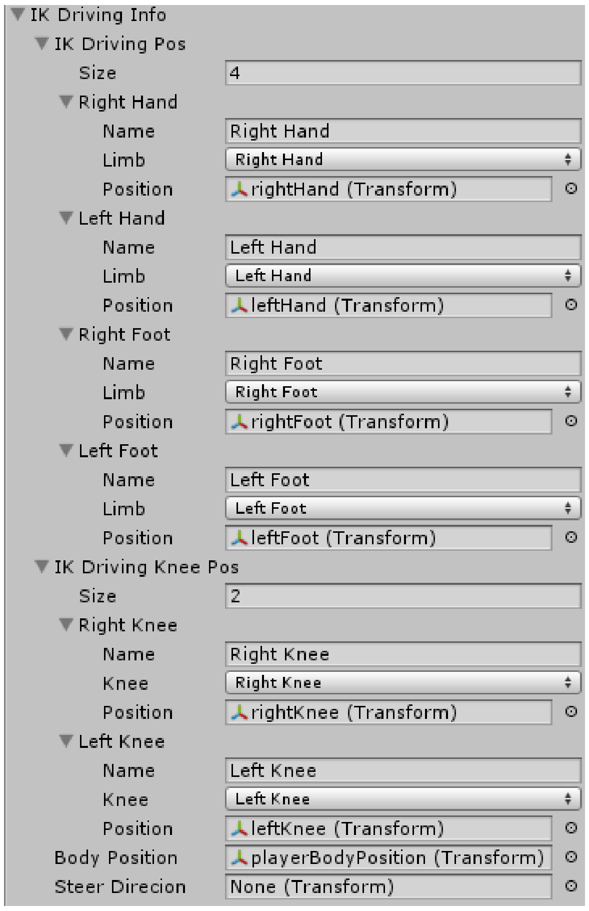
Like this, the positions are stored in a list which is sent to the ik system in the player to disable the player controller and set the limb positions and rotations.

Also, it is used when the player gets in and off from vehicle, to enable and disable player and vehicle, setting the current camera pivot control in player or vehicle and moving if from one pivot to another.



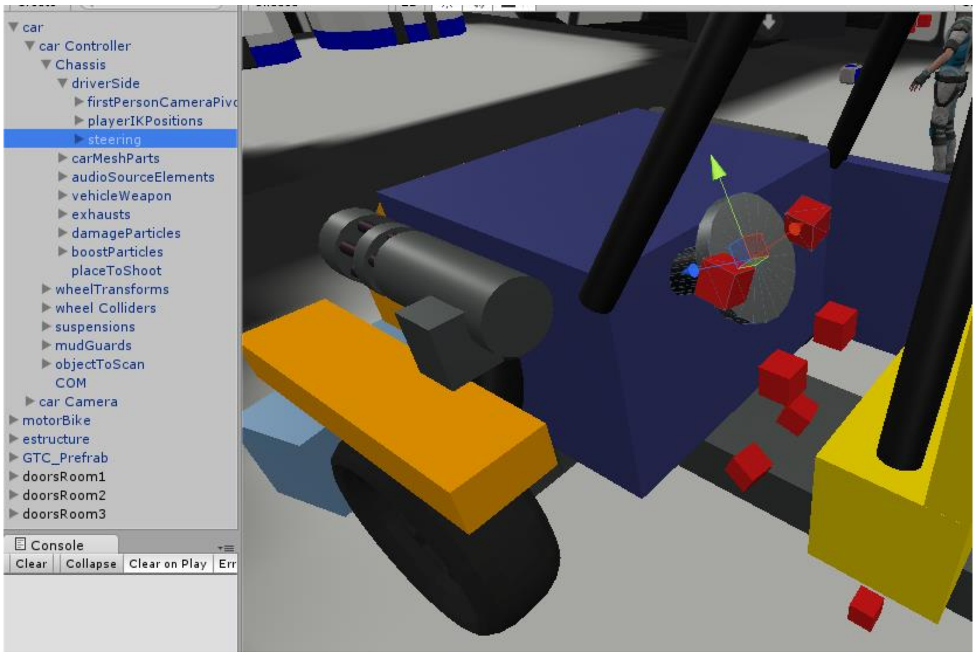
IK Driving System (Script)
Script: IKDrivingSystem
IK Driving Info
Eject Player When Destroyed: ☐
Ejecting Player Force: 20

- **Eject player when destroyed**, if the vehicle is destroyed while the player is driving, the option ejects the player from vehicle without hurting him, or the player dies when the vehicle explodes.
- **Ejecting player force**, is the force applied to the player in his local up transform direction when he is ejected from the vehicle.


IK Driving Info
IK Driving Pos
Size 4
Right Hand
Name Right Hand
Limb Right Hand
Position rightHand (Transform)
Left Hand
Name Left Hand
Limb Left Hand
Position leftHand (Transform)
Right Foot
Name Right Foot
Limb Right Foot
Position rightFoot (Transform)
Left Foot
Name Left Foot
Limb Left Foot
Position leftFoot (Transform)
IK Driving Knee Pos
Size 2
Right Knee
Name Right Knee
Knee Right Knee
Position rightKnee (Transform)
Left Knee
Name Left Knee
Knee Left Knee
Position leftKnee (Transform)
Body Position playerBodyPosition (Transform)
Steer Direcion None (Transform)

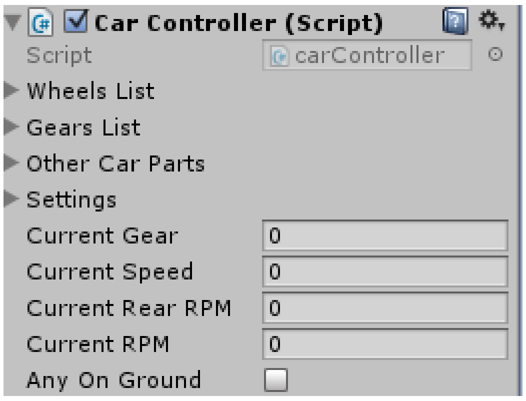
The IK driving info it allows to set the ik goals of the player's body (hands and feets) and the knees as well using transforms gameObjects inside every vehicle. Finally, it stores the player's position in the vehicle and the steer direction if the vehicle is a motorbike, to rotate the upper body too.

In this picture you can see all the IK positions used for the car, showing how it is organized in the hierarchy. The cube in the bottom is the player's position. The hands positions are inside the steering wheel so, when it is rotated, the hands follow that rotation.



CAR CONTROLLER, it controls every wheel collider stored in the wheel list, checking the distance to the ground and applying force to every of them, allowing to move the vehicle. Also, it control the input of the keyboard and touch controls from the input manager values, to drive the vehicle, allowing to move to left, right, back and forward, rotate the steering wheel, use the boost, jump and break.

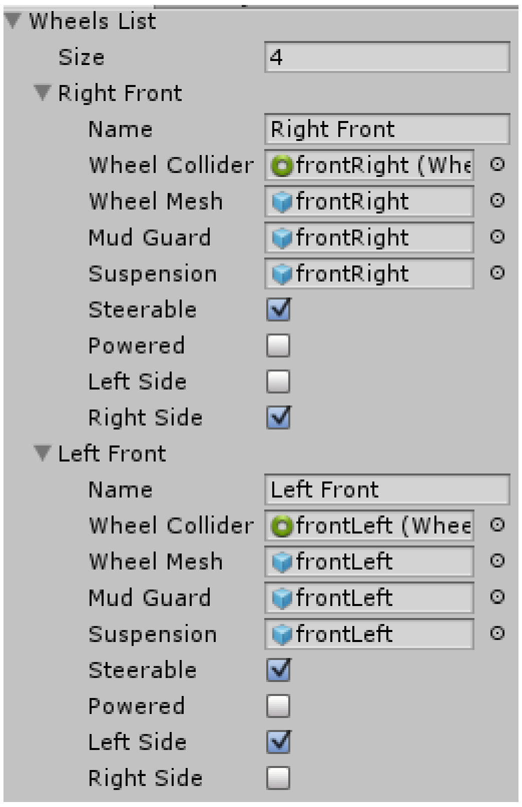
Also, it controls the boost, exhaust and slip particles, according to the input and movement values, and playing the sound effects to that states.



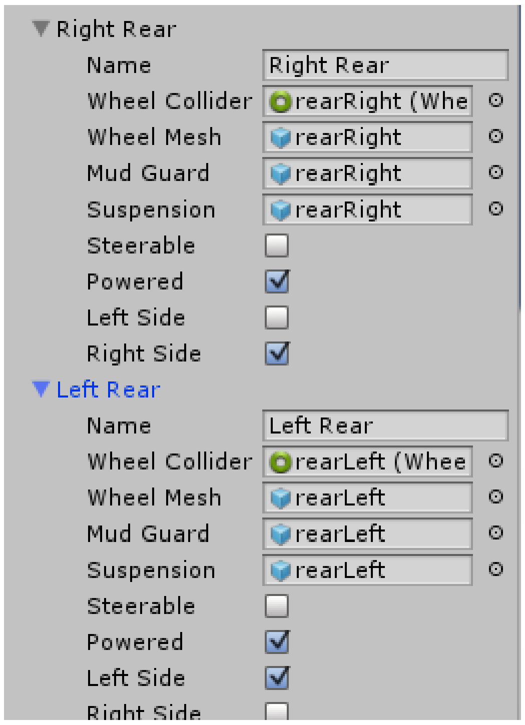
Car Controller (Script)

Script	carController
Wheels List	
Gears List	
Other Car Parts	
Settings	
Current Gear	0
Current Speed	0
Current Rear RPM	0
Current RPM	0
Any On Ground	☐

The float values in the inspector are to see the current values of the vehicle like current gear, speed, rpm in the rear part of vehicle and if there is at least a wheel in the ground.

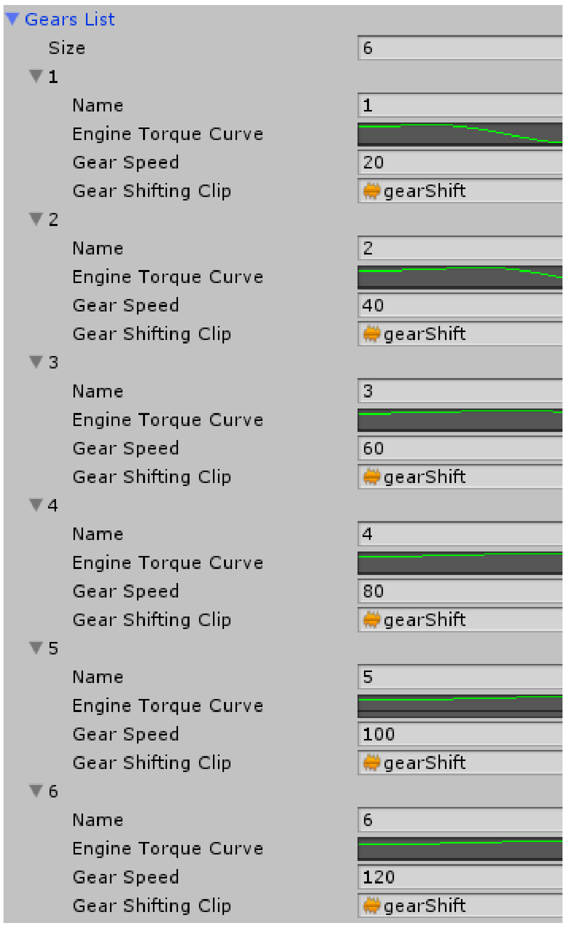


▼ Wheels List	
Size	4
▼ Right Front	
Name	Right Front
Wheel Collider	frontRight (Whe
Wheel Mesh	frontRight
Mud Guard	frontRight
Suspension	frontRight
Steerable	✓
Powered	☐
Left Side	☐
Right Side	✓
▼ Left Front	
Name	Left Front
Wheel Collider	frontLeft (Whee
Wheel Mesh	frontLeft
Mud Guard	frontLeft
Suspension	frontLeft
Steerable	✓
Powered	☐
Left Side	✓
Right Side	☐



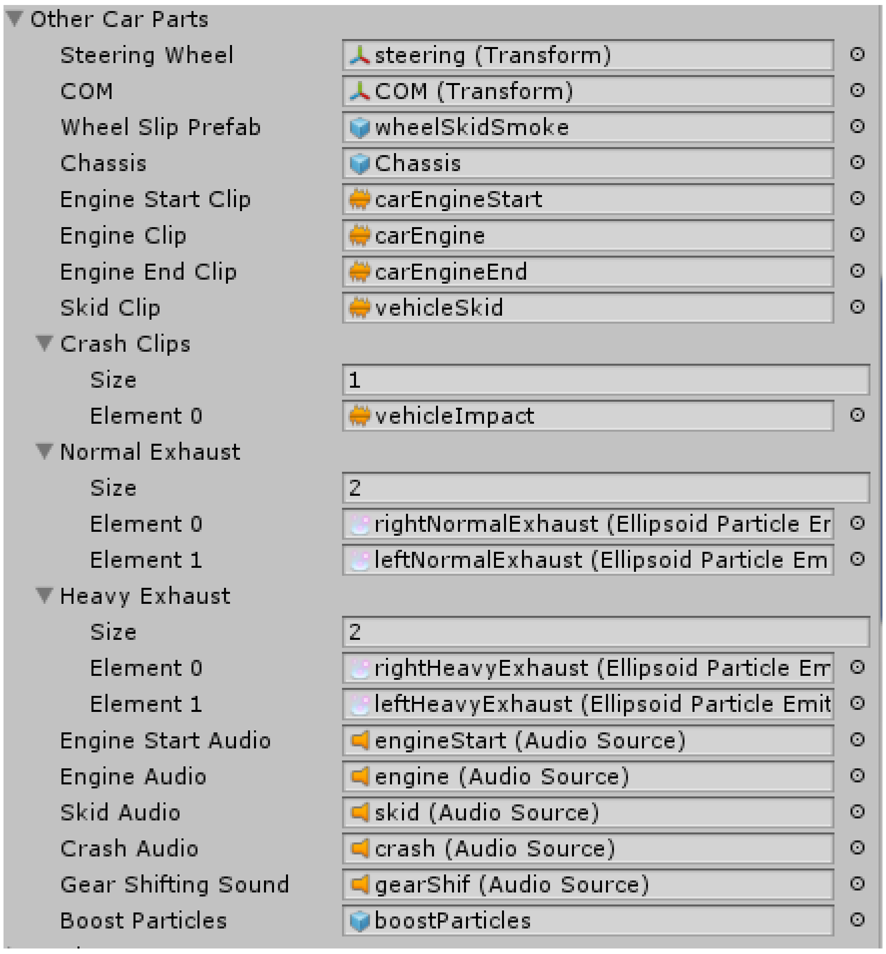
▼ Right Rear	
Name	Right Rear
Wheel Collider	rearRight (Whe
Wheel Mesh	rearRight
Mud Guard	rearRight
Suspension	rearRight
Steerable	☐
Powered	✓
Left Side	☐
Right Side	✓
▼ Left Rear	
Name	Left Rear
Wheel Collider	rearLeft (Whee
Wheel Mesh	rearLeft
Mud Guard	rearLeft
Suspension	rearLeft
Steerable	☐
Powered	✓
Left Side	✓
Right Side	☐

The wheel list allows to configure the wheel colliders, wheel meshes, suspensions and mud guards for every wheel of the vehicle. Also, it is set if the wheel is steerable, powered and the side of the vehicle. For version 2.3.5 it is better has 2 rear and 2 front wheels for a correct control of vehicle. For next update the system will allow to have as many wheels as you want,



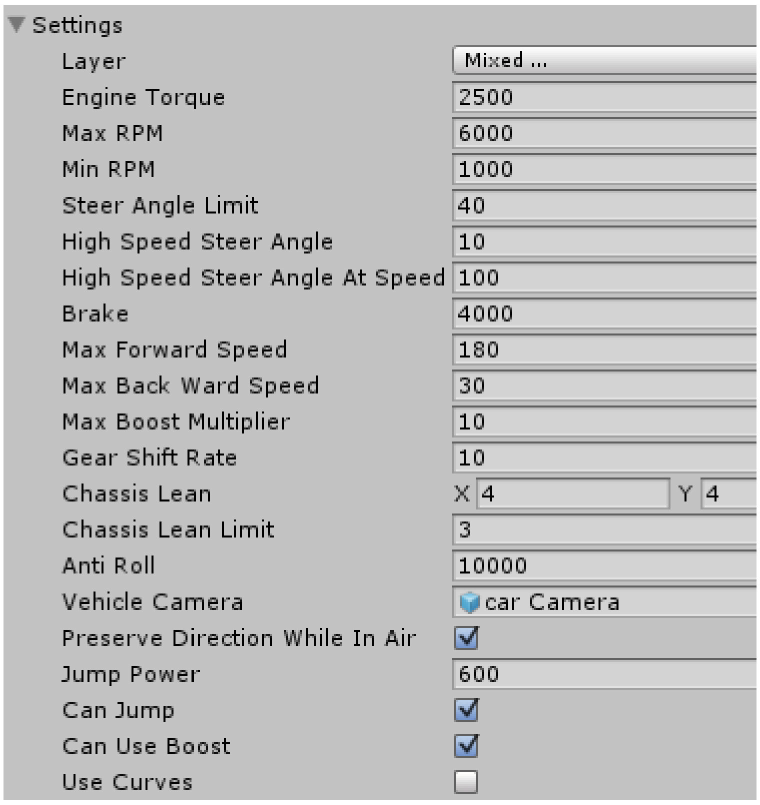
```
▼ Gears List
  Size                      6
  ▼ 1
    Name                    1
    Engine Torque Curve
    Gear Speed              20
    Gear Shifting Clip      gearShift
  ▼ 2
    Name                    2
    Engine Torque Curve
    Gear Speed              40
    Gear Shifting Clip      gearShift
  ▼ 3
    Name                    3
    Engine Torque Curve
    Gear Speed              60
    Gear Shifting Clip      gearShift
  ▼ 4
    Name                    4
    Engine Torque Curve
    Gear Speed              80
    Gear Shifting Clip      gearShift
  ▼ 5
    Name                    5
    Engine Torque Curve
    Gear Speed              100
    Gear Shifting Clip      gearShift
  ▼ 6
    Name                    6
    Engine Torque Curve
    Gear Speed              120
    Gear Shifting Clip      gearShift
```

In the vehicle a list of gears can be configured, to change them according to the current speed. Every gear has a sound effect and a speed to change to the next gear, along with an animation curve, for the speed increase.



Other Car Parts	
Steering Wheel	steering (Transform)
COM	COM (Transform)
Wheel Slip Prefab	wheelSkidSmoke
Chassis	Chassis
Engine Start Clip	carEngineStart
Engine Clip	carEngine
Engine End Clip	carEngineEnd
Skid Clip	vehicleSkid
Crash Clips	
Size	1
Element 0	vehicleImpact
Normal Exhaust	
Size	2
Element 0	rightNormalExhaust (Ellipsoid Particle Er
Element 1	leftNormalExhaust (Ellipsoid Particle Em
Heavy Exhaust	
Size	2
Element 0	rightHeavyExhaust (Ellipsoid Particle Em
Element 1	leftHeavyExhaust (Ellipsoid Particle Emit
Engine Start Audio	engineStart (Audio Source)
Engine Audio	engine (Audio Source)
Skid Audio	skid (Audio Source)
Crash Audio	crash (Audio Source)
Gear Shifting Sound	gearShif (Audio Source)
Boost Particles	boostParticles

- **Steering wheel**, the steering wheel of the vehicle which rotates according to the steering direction of the vehicle.
- **COM**, center of mass of the vehicle, used for the rigidbody.
- **Wheel slip prefab**, the slip particles instantiated in every wheel.
- **Chassis**, the chassis of vehicle which contains every important part of the vehicle, like meshes, particles, wheels parts and colliders.
- **Engine and skid clips**, audio clips for every part of the engine and the skid state.
- **Crash clips**, a list of crash audio clips used when the vehicle collides with another object.
- **Normal and heavy exhaust**, a list of exhaust particles used for the vehicle.
- **Engine start, engine, skid and gear shifting sounds**, the different audio sources for every sound in the vehicle.
- **Boost particles**, an empty gameObject that contains any amount of boost particles.



Settings	
Layer	Mixed ...
Engine Torque	2500
Max RPM	6000
Min RPM	1000
Steer Angle Limit	40
High Speed Steer Angle	10
High Speed Steer Angle At Speed	100
Brake	4000
Max Forward Speed	180
Max Back Ward Speed	30
Max Boost Multiplier	10
Gear Shift Rate	10
Chassis Lean	X 4 Y 4
Chassis Lean Limit	3
Anti Roll	10000
Vehicle Camera	car Camera
Preserve Direction While In Air	✓
Jump Power	600
Can Jump	✓
Can Use Boost	✓
Use Curves	☐

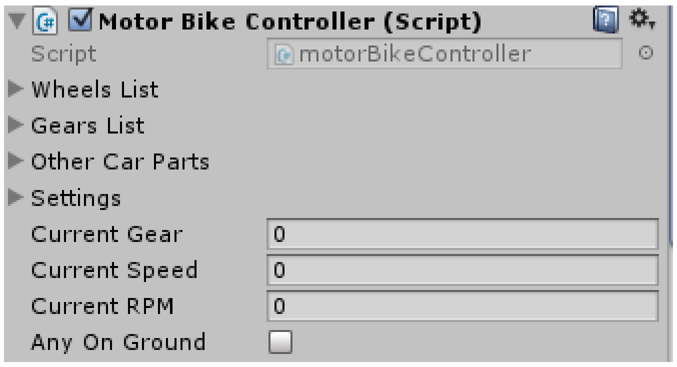
- **Layer**, the layer used to use a raycast in every wheel collider to check the distance to the ground.
- **Engine torque**, the amount of motor torque applied to every powered wheel.
- **Max rpm**, the max rpm reached by the vehicle engine.
- **Min rpm**, the min rpm applied to the vehicle engine.
- **Steer angle limit**, the max amount of rotation for the steerable wheels
- **High speed steer angle**, once this speed is reached, the wheels will rotate a certain amount of degrees
- **High speed steer angle at speed**, the maximum rotation applied to steerable wheels once the vehicle speed is higher than the high speed steer angle value.
- **Brake**, the amount of brake torque applied to every wheel when the vehicle breaks.
- **Max forward speed**, the max amount of speed reached by the vehicle in forward direction
- **Max backward speed**, the max amount of speed reached by the vehicle when it moves backward.
- **Max boost multiplier**, a multiplier applied to the motor torque when the boost is enabled
- **Gear shift rate**, a multiplier for the gear shift used for the current rpm of the vehicle.
- **Chassis lean**, the amount of rotation applied to the chassis while the vehicle is shifting a gear.
- **Chassis lean limit**, the clamp value for the chassis lean.
- **Anti roll**, a multiplier to adhere the wheel colliders to the ground, for a better control
- **Vehicle camera**, the camera used for this vehicle.
- **Preserve direction in air**, if the vehicle is in the air a certain amount of time, the vehicle is rotated in the velocity direction.
- **Jump power**, the amount of power for the vehicle jump.

- **Can jump**, the vehicle can jump or not.
- **Can use boost**, the vehicle can use the boost or not.
- **Use curve**, if true, the curve configured in the gear list is used for the speed acceleration.

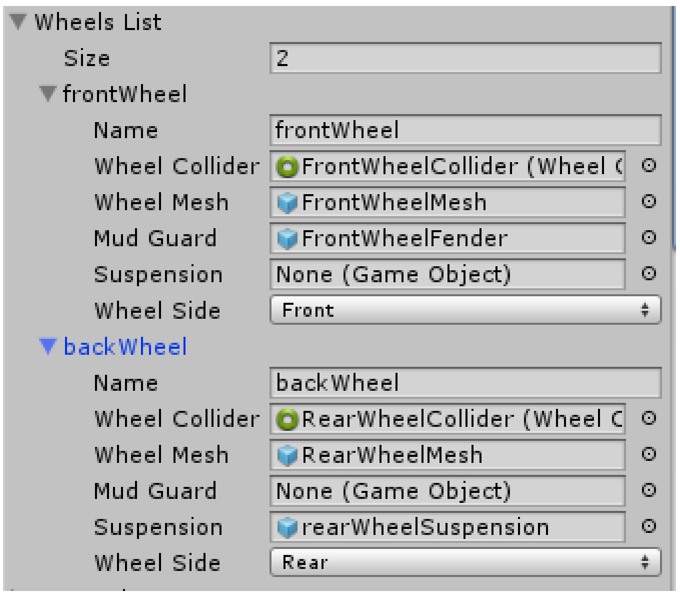
MOTOR BIKE CONTROLLER, the motorbike has almost the same working that the car and most of its attributes. The only difference is the code that checks every wheel state and that the motorbike is balanced in every moment in its z local axis in any surface, applying rotation to the rigidbody, to stabilize it.

Also, when the motorbike is in the air a certain amount of time, it is balanced in its x local axis, to have a better control of it.

The rest of the code and inspector it almost equal to the car, with sounds, particles, input values and steering values.



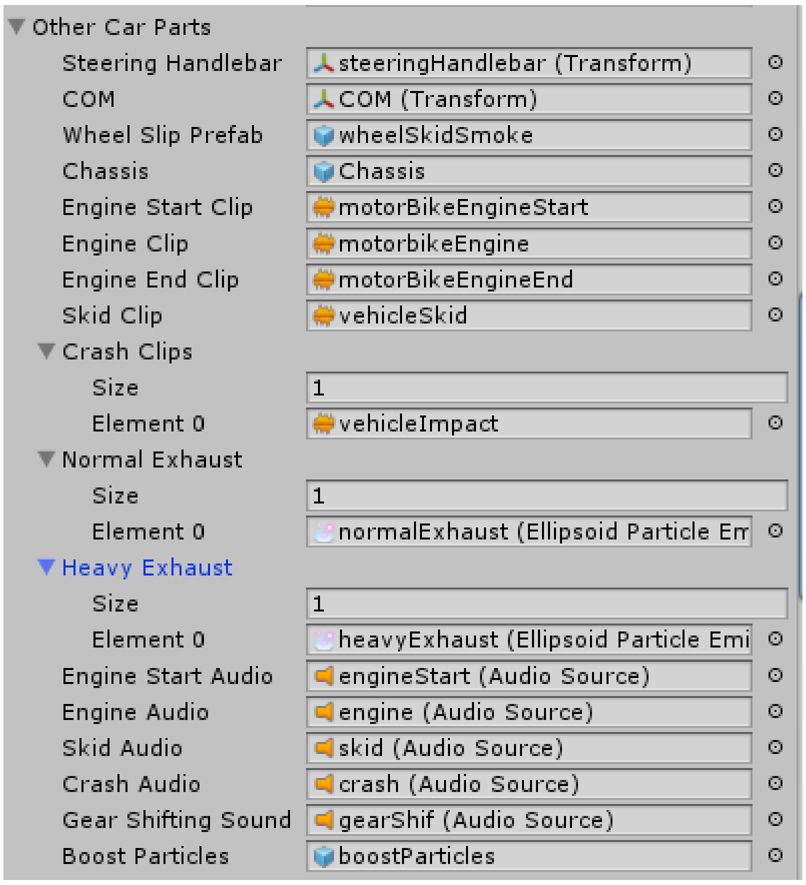
Motor Bike Controller (Script)	
Script	motorBikeController
Wheels List	
Gears List	
Other Car Parts	
Settings	
Current Gear	0
Current Speed	0
Current RPM	0
Any On Ground	☐



Wheels List	
Size	2
frontWheel	
Name	frontWheel
Wheel Collider	FrontWheelCollider (Wheel C
Wheel Mesh	FrontWheelMesh
Mud Guard	FrontWheelFender
Suspension	None (Game Object)
Wheel Side	Front
backWheel	
Name	backWheel
Wheel Collider	RearWheelCollider (Wheel C
Wheel Mesh	RearWheelMesh
Mud Guard	None (Game Object)
Suspension	rearWheelSuspension
Wheel Side	Rear

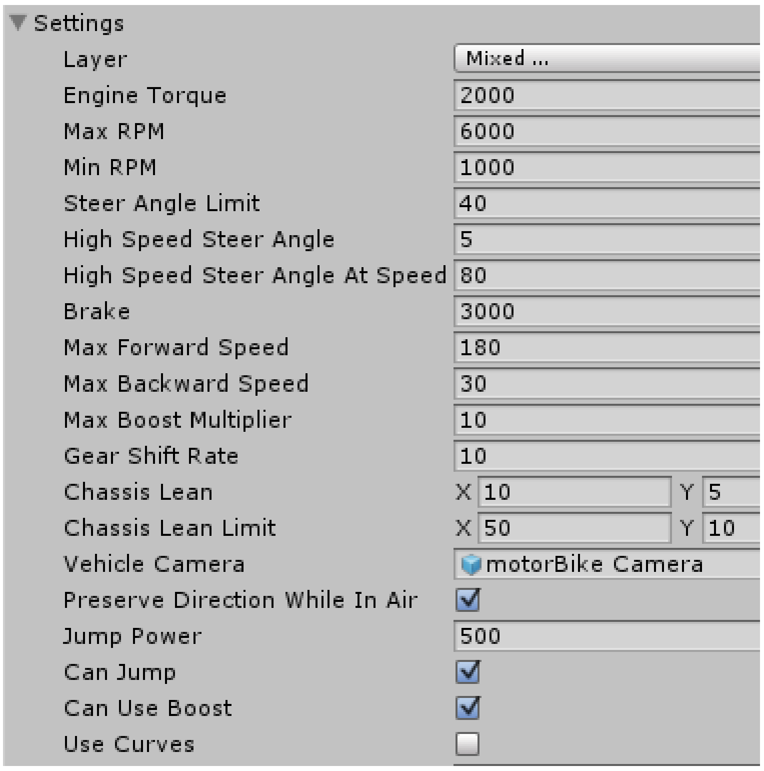
Like in the car, it is recommended to use only two wheels in motorbikes, one in the front and another in the back.

Like in cars, the motorbike can has suspensions and mudguards and set if the wheel is a front or rear type.



Other Car Parts	
Steering Handlebar	steeringHandlebar (Transform)
COM	COM (Transform)
Wheel Slip Prefab	wheelSkidSmoke
Chassis	Chassis
Engine Start Clip	motorBikeEngineStart
Engine Clip	motorbikeEngine
Engine End Clip	motorBikeEngineEnd
Skid Clip	vehicleSkid
Crash Clips	
Size	1
Element 0	vehicleImpact
Normal Exhaust	
Size	1
Element 0	normalExhaust (Ellipsoid Particle Em
Heavy Exhaust	
Size	1
Element 0	heavyExhaust (Ellipsoid Particle Emi
Engine Start Audio	engineStart (Audio Source)
Engine Audio	engine (Audio Source)
Skid Audio	skid (Audio Source)
Crash Audio	crash (Audio Source)
Gear Shifting Sound	gearShif (Audio Source)
Boost Particles	boostParticles

This part of the inspector has the same parameters that cars.



Settings	
Layer	Mixed ...
Engine Torque	2000
Max RPM	6000
Min RPM	1000
Steer Angle Limit	40
High Speed Steer Angle	5
High Speed Steer Angle At Speed	80
Brake	3000
Max Forward Speed	180
Max Backward Speed	30
Max Boost Multiplier	10
Gear Shift Rate	10
Chassis Lean	X 10 Y 5
Chassis Lean Limit	X 50 Y 10
Vehicle Camera	motorBike Camera
Preserve Direction While In Air	✓
Jump Power	500
Can Jump	✓
Can Use Boost	✓
Use Curves	☐

Finally, the settings are almost equal to cars, without the anti roll value in it.

The reset of the scripts for vehicles are the same, being this and car controller the only different between vehicles.

VEHICLE CAMERA CONTROLLER, every vehicle has its own camera system, similar to the player. It has a parent which follows the vehicle position a pivot to rotate in the local Y axis and an empty gameObject child to rotate in the local X axis.

When the player gets in to the vehicle, the main camera of the player is set as a child of the empty gameObject inside the pivot of the vehicle camera, placed in the local position 0,0,0. So in this state the camera player is not used, instead the vehicle camera controller is used having inside of it the main camera of the game.

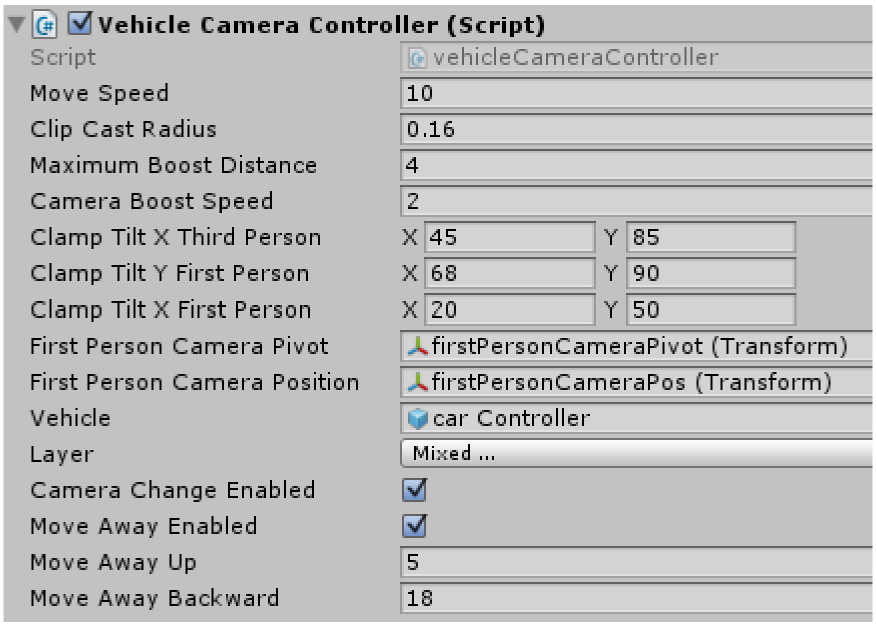
When the player gets off from the vehicle, the player camera gets the current vehicle rotation to get a smooth transition of the camera position and the camera rotation of the vehicle is reset, so every time the player gets in to the vehicle, the camera is placed in the back of the vehicle.

Like the player camera, the vehicle camera controller has a collision detection and it gets the input axis values from input manager, receiving the mouse or the touch joystick input from it.

This camera can has a third and a first person camera positon, so the player can change between both views, setting a rotation limit in both views. The script checks if the player is in third or first person, so when the player is in the vehicle, the camera position and rotation is set according to that value.

When the boost is being using, the camera move away from the vehicle and a boost particles are enabled. If the car energy reaches 0 or the boost is disabled, the camera backs to its position and the particles are disabled.

Finally, the camera can be moved away from the vehicle like the player camera does, pressing the default button left ctrl to move away or to set back.



Vehicle Camera Controller (Script) ✓

Script	vehicleCameraController
Move Speed	10
Clip Cast Radius	0.16
Maximum Boost Distance	4
Camera Boost Speed	2
Clamp Tilt X Third Person	X 45 Y 85
Clamp Tilt Y First Person	X 68 Y 90
Clamp Tilt X First Person	X 20 Y 50
First Person Camera Pivot	firstPersonCameraPivot (Transform)
First Person Camera Position	firstPersonCameraPos (Transform)
Vehicle	car Controller
Layer	Mixed ...
Camera Change Enabled	✓
Move Away Enabled	✓
Move Away Up	5
Move Away Backward	18

- **Move speed**, the move rotation speed in the camera.
- **Clip cast radius**, the sphere radius of the camera to check collisions.

- **Maximum boost distance**, maximum extra distance applied to the camera when the boost is being used.
- **Camera boost speed**, speed of the camera moving away when the boost is being used.
- **Clamp tilt x and y in first and third person**, the rotation limit in first and third person.
- **First person camera pivot and position**, the first person position and pivot of the camera.
- **Vehicle**, the vehicle using this camera.
- **Layer**, the layer used to check collisions between the camera and a surface.
- **Camera change enabled**, the camera can change between third and first person.
- **Move away enabled**, the camera can be moved away and set back to its regular position.
- **Move away up**, the amount extra of distance applied in local up direction when the camera is moved away.
- **Move away backward**, the amount extra of distance applied in local backward direction when the camera is moved away.

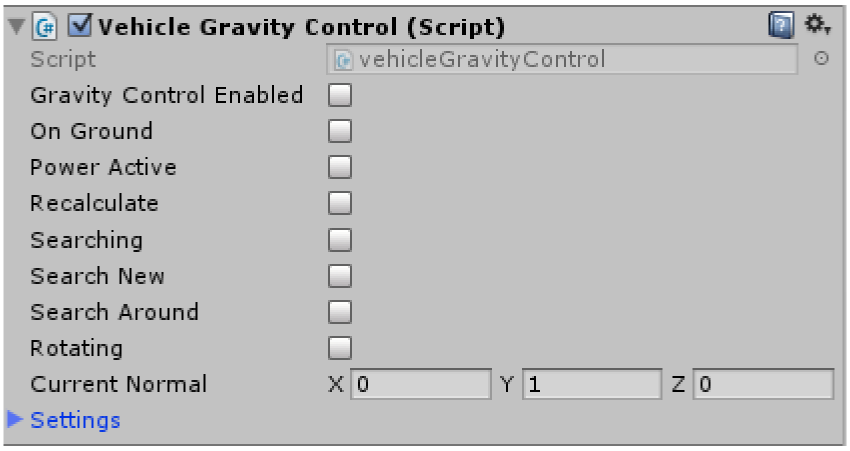
VEHICLE GRAVITY CONTROL, this component allows you to change the gravity direction in a similar way to the player. When you use the gravity control, the vehicle is launched directly in the camera direction, instead of floating in the air. This script also is used to apply force in the down local direction of the vehicle, due to its rigidbody has no gravity.

Like the player, when the vehicle is searching a surface, the vehicle can be moved to left, right, up and down and use the run button to increase the movement velocity.

When the player get off from the vehicle in a different gravity, the player is set automatically to that new gravity. This only happens if the vehicle is on the ground of its gravity, if the player gets off from vehicle in air, the vehicle gravity is reset and it falls to the regular gravity ground.

When the vehicles are searching a new gravity surface, the input doesn't move the wheels.

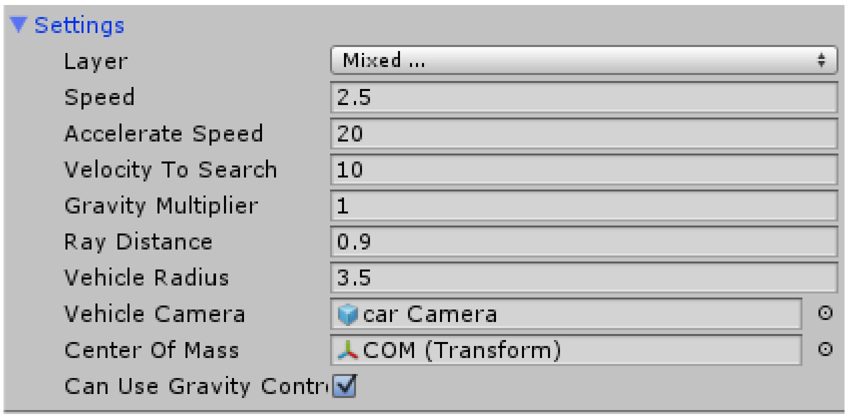
The rest of the system works very similar to the player change gravity component.



Vehicle Gravity Control (Script)

Script	vehicleGravityControl
Gravity Control Enabled	☐
On Ground	☐
Power Active	☐
Recalculate	☐
Searching	☐
Search New	☐
Search Around	☐
Rotating	☐
Current Normal	X 0 Y 1 Z 0
▶ Settings	

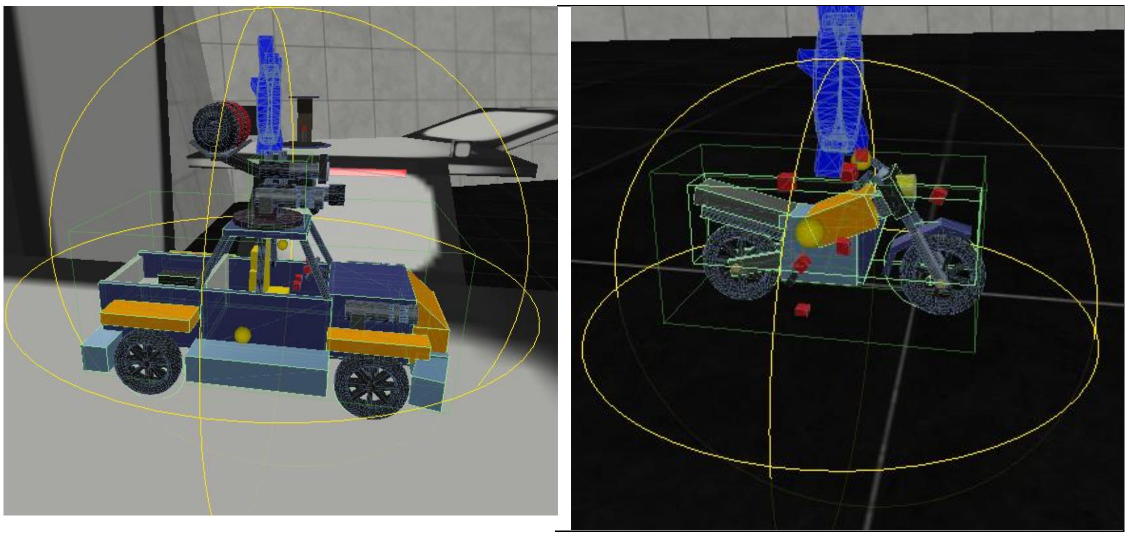
The Boolean values in the inspector are used to show the current state of the vehicle gravity.



▼ Settings	
Layer	Mixed ...
Speed	2.5
Accelerate Speed	20
Velocity To Search	10
Gravity Multiplier	1
Ray Distance	0.9
Vehicle Radius	3.5
Vehicle Camera	car Camera
Center Of Mass	COM (Transform)
Can Use Gravity Contr	✓

- **Layer**, layermask used to check with a raycast if the player is of the ground or not, to apply force in down direction and to check new surfaces.
- **Speed**, the speed multiplier applied to the vehicle when it searches a new surface.
- **Accelerate speed**, the acceleration multiplier when the player increases the movement velocity in the air, searching a new surface.
- **Velocity to search**, if the current gravity is different from the regular, when the player is in the air and reaches this velocity, enable the raycast in the local down direction to searches a new surface to rotate the vehicle to a new gravity direction
- **Gravity multiplier**, the gravity multiplier to apply force to the vehicle in the down direction.
- **Ray distance**, the distance of the raycast.
- **Vehicle radius**, the size of the raycast used to search a new surface when the gravity control is enabled. The bigger the vehicle, the bigger the radius.
- **Vehicle camera**, the camera used by the vehicle.
- **Center of mass**, center of mass (COM), of the vehicle.
- **Can use gravity control**, the vehicle can use the gravity control.

In this picture you can see the vehicle radius, according to the size of the vehicle.



VEHICLE HUD MANAGER, this script is similar to health. It manages the health, energy and ammo of a vehicle, setting their values in the vehicle HUD.

The health and energy can be regenerative, or increase by using pickups. This script has the function called by the using Devices System from the player, when the use button is pressed, to get in of off from the vehicle.

If the vehicle has not a weapon system, the ammo and weapon info is not showed in the HUD.

When the player wants to gets off from the car, it can be configured from what side of the car the player exits (right or left), checking in that side is not blocking by an object, so the player will get off in the other side, checking in the same way, so if both sides are blocked, the player won't get off.

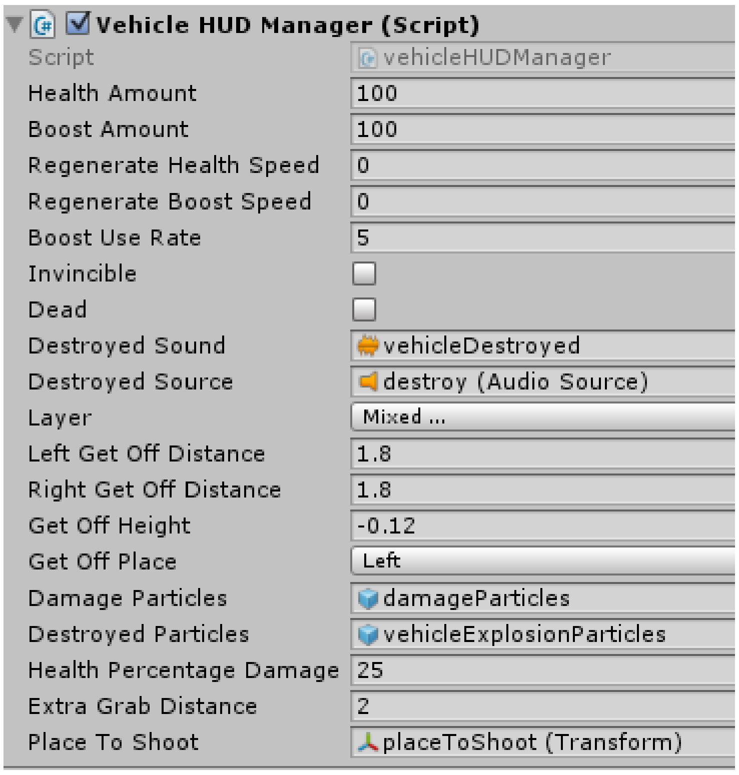
There are two yellow spheres which their positions are configured in the inspector. They are used to see the positions where a ray is used to check both sides, and the another ray is used to check the distance to the ground, so the player is placed (nicely) at the side of the vehicle.

This system allows to take damage by the any projectile fired by the player or the turrets, and if health is 0, the vehicle ejects the player and it is destroyed or the player dies. While the player is driving the vehicle, he doesn't receive damage, but once he is out of it, he receive it. Also, when he is driving the pickups are only grabbed by the vehicle to refill its parameters. Vehicles received damage when they collide with another vehicle.

If the health of the vehicle is lower than a certain percentage, fire and smoke particles are activated in the vehicle, to warning the player. If the health still decreasing, the fire will be bigger. If he gets health pickups, the fire will disappear.

You can add as many colliders (box or mesh) as you want (according to the vehicle meshes), which are used to check the damage received by every vehicle, so like this the damage detection is really accurate. For example, if you shoot a grenade to a car, every collider will receive the explosion, but the vehicle will only be damaged once, with the correct amount.

Finally, when the vehicle is destroyed, all vehicle components are disabled, a rigidbody and a box collider is attached to every mesh renderer, and after some seconds, the meshes start to vanish and once they alpha value is 0, the vehicle is destroyed.


Vehicle HUD Manager (Script)

Script	vehicleHUDManager
Health Amount	100
Boost Amount	100
Regenerate Health Speed	0
Regenerate Boost Speed	0
Boost Use Rate	5
Invincible	☐
Dead	☐
Destroyed Sound	vehicleDestroyed
Destroyed Source	destroy (Audio Source)
Layer	Mixed ...
Left Get Off Distance	1.8
Right Get Off Distance	1.8
Get Off Height	-0.12
Get Off Place	Left
Damage Particles	damageParticles
Destroyed Particles	vehicleExplosionParticles
Health Percentage Damage	25
Extra Grab Distance	2
Place To Shoot	placeToShoot (Transform)

- **Health and boost amount**, max and current amount of health and boost.
- **Regenerate health and boost speed**, speed to regenerate health and energy. If equal to 0, these values are not regenerated.
- **Boost use rate**, a multiplier to decrease the boost value while it is being used.
- **Invincible**, if true, the vehicle cannot be damaged.
- **Dead**, the vehicle health is 0.
- **Destroyed sound and source**, the clip and audio source played when the vehicle is destroyed.
- **Layer**, the layermask used to check for obstacles when the player gets off from the vehicle.
- **Left and right get off distance**, an offset to set the position where a sphere cast is used to check for any obstacle at the sides of the vehicle.
- **Get off height**, an offset for the height position of the sphere cast used to search obstacles when the player want to get off from the vehicle.
- **Get off place**, set if the player gets off from vehicle for the left or right side.
- **Damage particles**, the fire and smoke particles used when the vehicle health is lower than a certain percentage.
- **Destroyed particles**, the particles instantiated when the vehicle is destroyed.
- **Health percentage damage**, when the health is lower that this percentage, the fire and smoke damage particles are enabled and increased according to the damage received.
- **Extra grab distance**, an extra amount of distance added when the player gabs this vehicle.
- **Place to shoot**, the place where the place to shoot from the player will be placed in the vehicle, since the enemies still searching for player's collider, which is a trigger when he is driving.

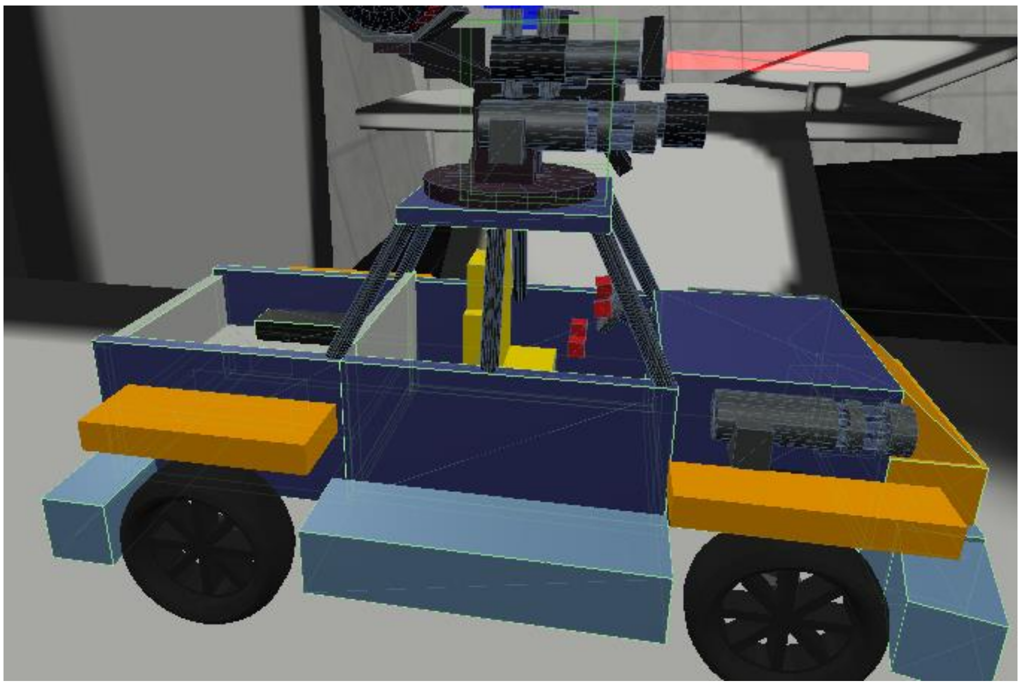
VEHICLE DAMAGE RECEIVER, when a projectile from enemies, friends or player hits a collider with this component, it send a message to vehicle HUD manager, to decrease the amount of health according to the damage produced by that projectile.

The only parameter of this script is a gameObject called vehicle which store the vehicle that has a collider with this component attached inside the vehicle. In the start function of vehicle HUD manager, it searches for every collider inside the vehicle with vehicle damage manager, setting the variable vehicle with that object.

Like this, you can add as many collider as you need and there is no need to configure the vehicle that owns them.

Also, this allow to have an accurate damage system for vehicles.

In this picture, every collider in the car has a damage receiver, to set the damage in the vehicle HUD manager component.



VEHICE WEAPON SYSTEM, it follows the camera rotation and a rotation limit can be configured. Like the powers, it is very easy to create and add new weapons, configuring its name, clip size, fire rate, shoot position, animation, sound, muzzle flash, ... The weapons can be changed using wheel mouse, number keys and two more buttons in the keyboard, for next and previous weapon. The ammo of every weapon can be infinite or use pickups to refill it. The current weapons in the vehicle are:

- Machine gun.
- Cannon.
- Laser.
- Homing missiles.
- Implosion grenades.
- Barrel launcher.
- Double machine gun.

- Seeker missile launcher which go to the closest enemy in front of the player.

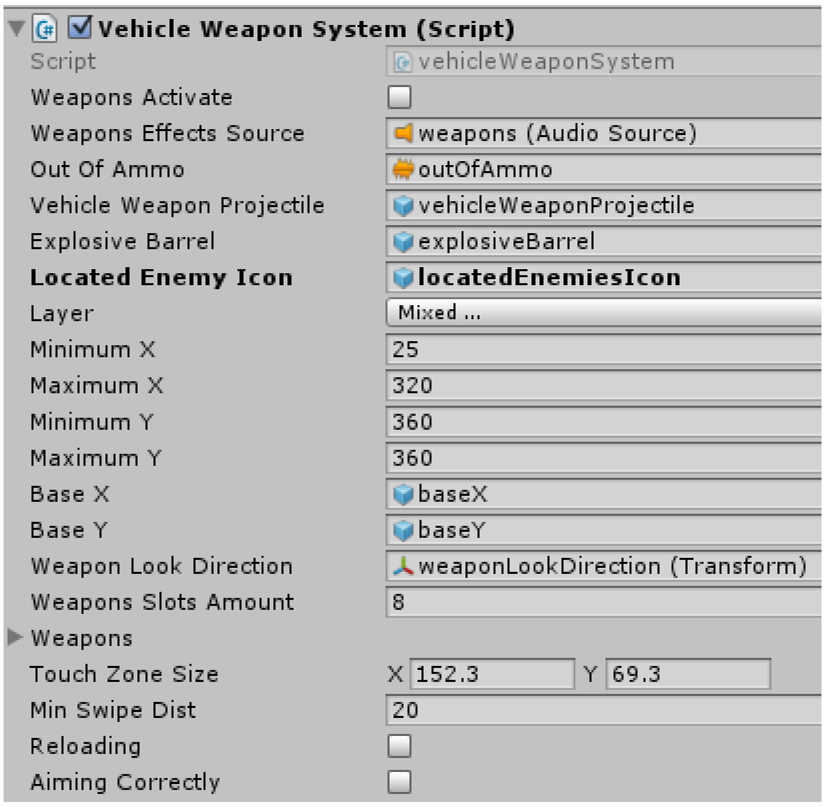
To make able that the weapon can use ammo pickups, it is only necessary to configure the amount of ammo and the name of the weapon that belongs. For example, to add a barrel launcher ammo, set the amount of ammo and the name “Barrel launcher” in it.

When the player/vehicle grabs a pickup, this will check if the amount of health, energy or ammo is filled or not, so the player/vehicle only will get the necessary objects to restore his state. In version 2.3, the player grabbed every pickup close to him. This is also applied when the player is close to more than one pickup, so if the player has 90/100 of health and he is close to two health pickups, he only will grab one of them.

The weapon of the vehicle checks when the rotation limit is reached, and in this case, the projectiles are fired in the weapon forward direction. This is made like that because the projectiles are fired in the camera direction, to have a perfect accuracy, so when the player is looking down, this avoid to shoot in a wrong direction.

The barrel launcher launch a barrel with a parable in the position where the camera aims. The barrel is launched applying a force to its rigidbody.

The weapon is fully controlled with the custom input manager from the asset, along with integrated touch controls for mobile. So the weapons can be changed by swiping in the right upper corner of the screen.

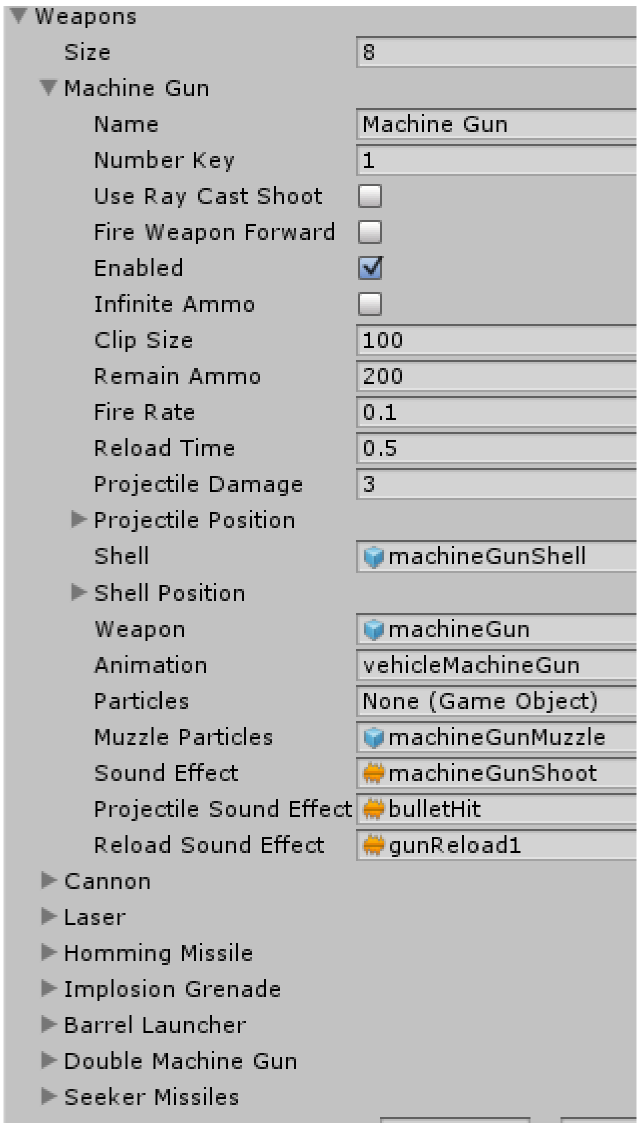


Vehicle Weapon System (Script)

Script	vehicleWeaponSystem
Weapons Activate	☐
Weapons Effects Source	weapons (Audio Source)
Out Of Ammo	outOfAmmo
Vehicle Weapon Projectile	vehicleWeaponProjectile
Explosive Barrel	explosiveBarrel
Located Enemy Icon	**locatedEnemiesIcon**
Layer	Mixed ...
Minimum X	25
Maximum X	320
Minimum Y	360
Maximum Y	360
Base X	baseX
Base Y	baseY
Weapon Look Direction	weaponLookDirection (Transform)
Weapons Slots Amount	8
Weapons	
Touch Zone Size	X 152.3 Y 69.3
Min Swipe Dist	20
Reloading	☐
Aiming Correctly	☐

- **Weapons activate**, if the vehicle is being driven, the weapons are activated.
- **Weapons effects source**, audio source for fire sounds.
- **Out of ammo**, audio clip for out of ammo.
- **Vehicle weapon projectile**, the prefab fired by every weapon.
- **Explosive barrel**, the explosive barrel prefab launched by the weapon.

- **Located enemy icon**, the icon used to show a locked enemy.
- **Layer**, the layer used for the raycast of the weapon to shoot at the aimed camera position correctly.
- **Minimum X and Y and maximum X and Y**, the rotation limit for both axis in the weapons turret.
- **Base X and Y**, the transform rotated in x and y axis local axis.
- **Weapon look direction**, an empty object which follows the camera rotation to get the rotation which will be applied to the weapon base.
- **Weapons slots amount**, the current number of weapons in the vehicle.
- **Weapons**, the list of weapons.
- **Touch zone size**, the touch zone size placed in the upper right side of the screen to change between weapons in a touch device.
- **Min swipe dist**, minimum distance to make a swipe, so the weapon is changed.
- **Reloading**, the current weapon is being reloading.
- **Aiming correctly**, the camera rotation is inside the limit of the weapon rotation.



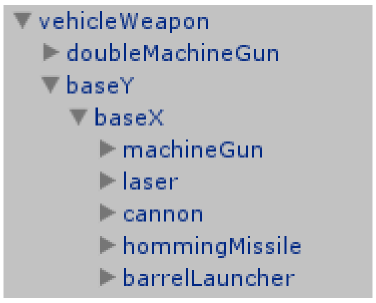
Weapons
Size: 8
Machine Gun
Name	Machine Gun
Number Key	1
Use Ray Cast Shoot	☐
Fire Weapon Forward	☐
Enabled	✓
Infinite Ammo	☐
Clip Size	100
Remain Ammo	200
Fire Rate	0.1
Reload Time	0.5
Projectile Damage	3
Projectile Position	
Shell	machineGunShell
Shell Position	
Weapon	machineGun
Animation	vehicleMachineGun
Particles	None (Game Object)
Muzzle Particles	machineGunMuzzle
Sound Effect	machineGunShoot
Projectile Sound Effect	bulletHit
Reload Sound Effect	gunReload1

Cannon
Laser
Homming Missile
Implosion Grenade
Barrel Launcher
Double Machine Gun
Seeker Missiles

Every weapon has a list of parameters to configure. To add or remove a weapon, change the size of the list.

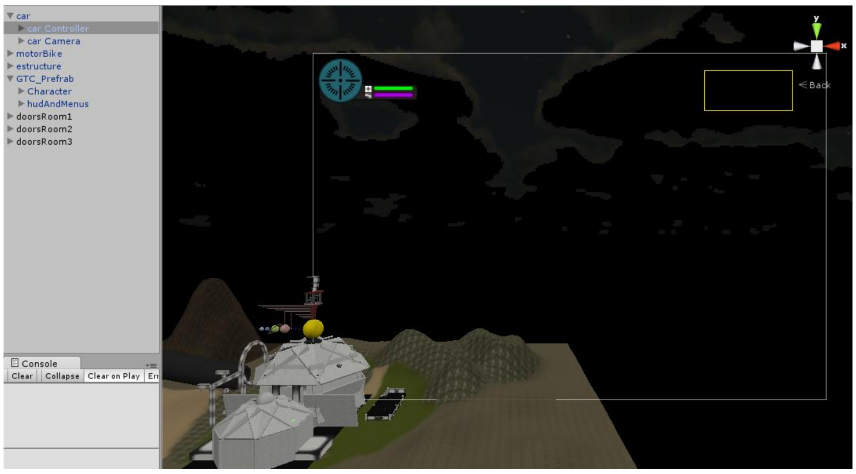
- **Name**, the name of that weapon.
- **Number key**, the key number used to choose that weapon.
- **Use ray cast shoot**, if true, when the weapon is fired, the projectiles are placed directly in the hit point got by a ray cast launched from the camera position. Else, the projectiles are launched applying velocity to their rigidbody.
- **Fire weapon forward**, if true the projectiles are fired in the forward direction of the weapon instead the camera direction.
- **Enabled**, if true that weapon can be used.
- **Infinite ammo**, if true, that weapon has infinite ammo.
- **Clip size**, the amount of ammo that every clip has, so when the weapon is reloaded, it gets a magazine of that size.
- **Remain ammo**, the rest of ammo to be used for that weapon.
- **Fire rate**, the amount of seconds between shoots.
- **Reload time**, when the clip is empty, is the time that takes to reload the weapon.
- **Projectile damage**, the damage applied to any object with a health component or a vehicle HUD manager.
- **Projectile position**, is a list of transform position, setting the number of projectiles fired at once. It is done to make for example a double weapon or a multiple missile launcher
- **Shell**, the empty shell of the projectile, it is not necessary to set it, for example a laser it hasn't It. A machine gun has it.
- **Shell position**, a list of transform position where the shells are created. If you set shell for the weapon, you have to set the same amount of projectile positions and shell positions, so for every projectile, a shell is dropped.
- **Weapon**, the parent gameObject that contains all the parts of that weapon, having the animation component of the weapon.
- **Animation**, the name of the animation for that weapon, it can be empty, it can be an empty field.
- **Particles**, the particles used for when the projectile fired hits a surface, like an explosion from the cannon. It can be empty.
- **Muzzle particles**, the particles played when the weapon is being fired.
- **Sound effect**, the clip played when the weapon fires.
- **Projectile sound effect**, the sound played when a projectile hits a surface.
- **Reload sound effect**, the reload clip played when the weapon is being reloaded.

If you don't want that the weapon rotates, you have to place it outside the base X and Y of the vehicle weapon. In this picture you can see where are placed every gun. Those who are rotated are inside the base X like the cannon. The double machine gun is in the same place all the time, so they are placed outside of both base transform.



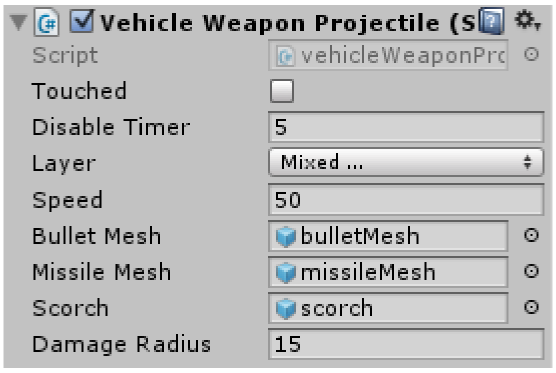
```
▼ vehicleWeapon
  ▶ doubleMachineGun
  ▼ baseY
    ▼ baseX
      ▶ machineGun
      ▶ laser
      ▶ cannon
      ▶ hommingMissile
      ▶ barrelLauncher
```

In this picture, if you select the vehicle with a weapon system in the hierarchy, and see the canvas HUD, you can see the gizmo that shows the touch zone position with a rectangle in the upper right corner of the screen, which can be configured in the weapon system inspector.



VEHICLE WEAPON PROJECTILE, the vehicle projectile are very similar that the used by the player, have two meshes, for bullets and missiles. A disable timer, a layer to check collisions, a scorch to place and a damage radius to search objects which can be damaged inside of it.

When a projectile is fired, the info of the current weapon is sent to the projectile to work correctly.



Vehicle Weapon Projectile (S

Script	vehicleWeaponPrc
Touched	☐
Disable Timer	5
Layer	Mixed ...
Speed	50
Bullet Mesh	bulletMesh
Missile Mesh	missileMesh
Scorch	scorch
Damage Radius	15

VEHICLE LASER, the laser are the same that for an enemy, enabled or disabling it according to if the current weapon used is that or not.

ALL THE SETTINGS ARE ALREADY CONFIGURATED IN THE DEMO AND IN THE PREFABS, JUST CHANGE THEM LIKE YOU WANT.

LAYERS AND TAGS

The player controller doesn't use the physics.gravity to work, so the general gravity in the scene isn't modified in any moment, the other rigidbodies are affected by the actual gravity values. So you can use that values as you want.

The asset uses different tags to work:

- **Player**, to set the main character of the game.
- **Box**, to set the objects that can be carried by the player.
- **Sphere**, to set the objects that can be circumnavigate by the player. It can be an object with any shape, this tag just make the player calculates his normal while he still walking in that object, allowing the player walk in a sphere for example.
- **Trail**, to set the trail renderer in the player. It is used by the script otherPowers, finding the objects with that tag, and activated them when the player runs. If there isn't any trail in the scene, the effect won't be activated. If you want to add to your model, just place them in its skeleton, in the places that you want.
- **Moving**, to set the objects that are moving or rotating. Like this the player is set as a child of that object, allowing him to walk in it in a similar way that a sphere object.
- **Black hole**, to set every black hole that the player fires. Every bullet of this type checks if there are any other black hole in the scene, to release the objects inside it and remove that black hole, so the new one can start.
- **Bullet**, to set every bullet that the player fires.
- **Slowing**, to set the objects that the player can slow down.
- **Friend**, to set the allies of the player, and the target that the ally turrets search along with the player.
- **Enemy**, to set the enemies of the player, and the target that the ally turrets search.
- **Device**, to set any usable device which the player can activate.
- **Fade**, just used to fade the walls of the room of the demo.

Also, the asset uses some layers:

- **Shield**, used for the player's shield.
- **Turrets**, used for the health component in the turrets.
- **Player**, used for the player's collider.
- **Radar**, used for the radar elements.
- **Wood, metal, snow and water**, used for the footsteps in mesh objects
- **Scanner**, used for the models above the scannable objects to make them visible in the scanner mode.
- **Vehicle**, used for the colliders in every vehicle with a vehicle damage receiver component.

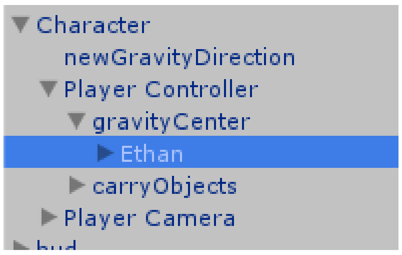
TUTORIALS

HOW TO SET YOUR OWN 3D MODEL

If you make a new scene, place the GTC_Prefab in the scene and make sure its position is 0,0,0, like this the HUD is showed correctly.

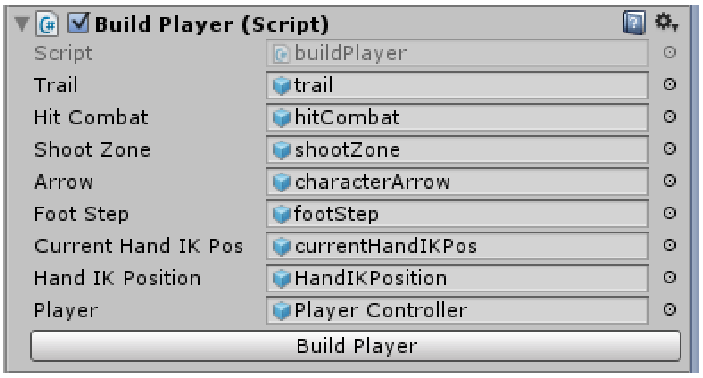
After that, move the player controller along player camera in the position that you need.

If you want to change the 3d model, just placed it inside the gravityCenter object and remove Ethan.



```
▼ Character
    newGravityDirection
  ▼ Player Controller
    ▼ gravityCenter
      ▶ Ethan
    ▶ carryObjects
  ▶ Player Camera
```

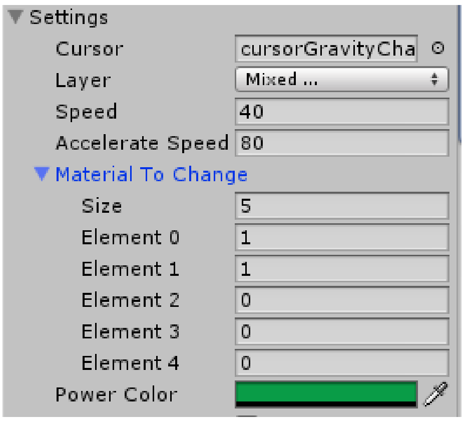
Then, go to the Character object in the hierarchy and press Build player. The character is placed in its position and all the objects are set inside character bones.



Build Player (Script)

Script	buildPlayer
Trail	trail
Hit Combat	hitCombat
Shoot Zone	shootZone
Arrow	characterArrow
Foot Step	footStep
Current Hand IK Pos	currentHandIKPos
Hand IK Position	HandIKPosition
Player	Player Controller

In changeGravity script set the meshCharacter parameter with the SkinnedMeshRenderer of your model in the settings of the script, and in Material to change set to 1 those you want to change. This settings is just if you want to change your model color, using the powerColor parameter. If you don't want to change any color, just don't configure this settings.

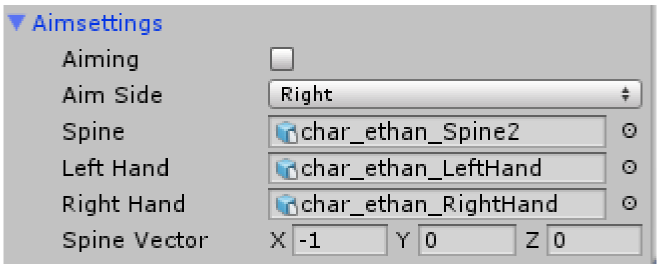


Settings

Cursor	cursorGravityCha
Layer	Mixed ...
Speed	40
Accelerate Speed	80
Material To Change	
Size	5
Element 0	1
Element 1	1
Element 2	0
Element 3	0
Element 4	0
Power Color	🟩

Then, place the red cubes in front of the player in the position that you want to place every hand in the aim mode.

If in aim mode the player’s rotation doesn’t follow correctly the camera, check that the spine object is the correct part of the player’s skeleton in other powers inspector.

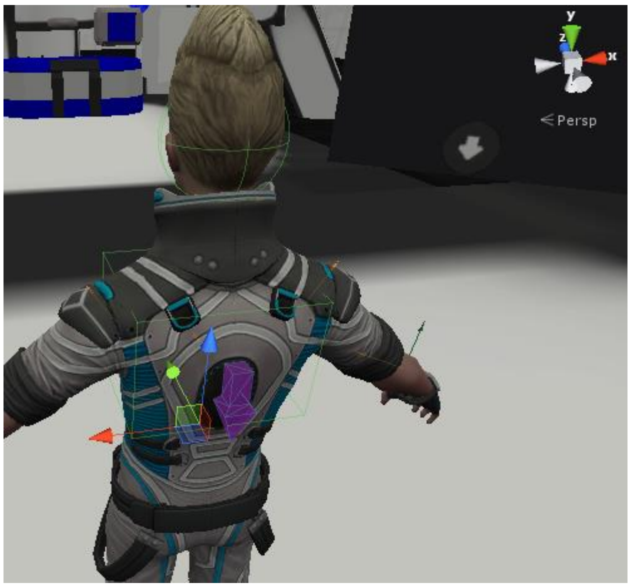


Aimsettings	
Aiming	☐
Aim Side	Right
Spine	char_ethan_Spine2
Left Hand	char_ethan_LeftHand
Right Hand	char_ethan_RightHand
Spine Vector	X -1 Y 0 Z 0

Then, check the spine local axis.

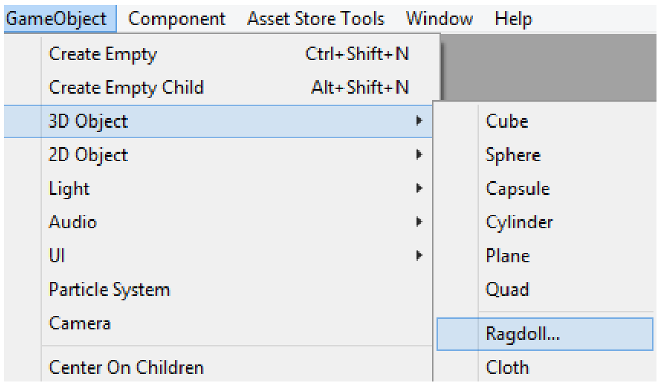
Since, every skeleton has its own rotations and axis directions, set in this vector the correct X axis in the spine of your player, to rotate the body of the player in the camera direction in aim mode.

In the image below, you can see how the Global X axis in the spine of the player aims in the negative X axis, so the rotation is made in that direction, being the vector (-1,0,0). Other models would have (0,0,-1) for example. Just check the spine axis in your model and changes this value in spine vector in the inspector.



If you want to use other model to the player and use a ragdoll when the player dies, you have two options:

- **Create a ragdoll with the wizard menu of unity**, setting every bone in the correct field. The script will detect if the character has a ragdoll or not.



- **Use the ragdollBuilder inspector in the Character gameObject**, pressing build ragdoll button, making a procedural ragdoll very similar to the one made by the wizard.

It is not mandatory to do this to be able to use this asset, just in case you want to use a ragdoll instead a death animation.

ADDITIONAL INFORMATION

ABOUT RAGDOLL

IMPORTANT: if you use the ragdoll system, and when the player get up, any bone inside it is deformed, just set the layer of that bone to Ignore Raycast. The face is the only part that seems to be affected, at least in the Ethan model. This is due to every model has different rotations in his bones, and the transition from the ragdoll to the animator rotates the bones in the direction of the get up animation, so to avoid any issue, just set the layer of the bones.

Also, there is a button to test the ragdoll, without the need that an enemy kills the player. Press M to activate the ragdoll.

ABOUT COMBAT SYSTEM

The close combat animations (kick and punches) are not included in this asset, but the system that use them. You can find those combat animations and more in the free asset Taichi Character pack, which was used to test this system. To use it, just set the animations in the combat layer of the animator.

SOUNDS EFFECTS

Some of the sounds effects in the asset have been obtained from a free sounds effects page: http://www.freesfx.co.uk.

SUGGESTIONS, ISSUES OR PROBLEMS

Asset YouTube Channel: https://www.youtube.com/channel/UCs21at6NKl_ieSIbE_2unuA

In this YouTube channel will be upload different tutorial videos for every system of this asset, so check it out.

Asset forum: http://forum.unity3d.com/threads/gravity-twist-controler-control-gravity-at-will-new-version-2-0.351456/

If you have any doubt problem or suggestions, please send me an email: santimonti90@gmail.com