Complete Physics Platformer Kit

The Guidebook

Thank you for purchasing the Complete Physics Platformer Kit! I hope this will be a solid solution to both 3D and 2D platformers, as well as advanced physics interaction in your games. I've worked hard to make this kit: modular, self-explanatory, easy to set up and most importantly FUN! Allowing you to focus on making your game, instead of coding interaction systems; to me this is not just a service to provide, but a way for me to give back to indie development which has been so kind to me.

If you have any questions or suggestions please reach me at: greganims1@gmail.com

Contents

1. Setup

There are 3 things to do for basic setup:

1. **Change gravity to -25 in Y axis** This is for the demo scene and pre-tweaked variables for jumping/throwing. You can use any gravity you like though. (edit > project settings > physics)
2. **Add a button 'Grab' and an axis 'CamHorizontal' to Inputs.** (edit > project settings > inputs)
3. Check the tags: Coin, Water, Pickup, Pushable, MovingPlatform and Waypoint have been added to your project

2. Things to know

- **Any trigger volumes should be set to: ignore raycast layer.** Why? Because the throwing script uses a sphereCast to check there is nothing above the players head

before pickup up an object. It is set to ignore layer index 2 (ignore raycast by default). Also the camera avoidance system raycasts from the player to the camera to check if the view is obscured.

- To make an object that can be carried/thrown by player, give it tag: **Pickup**, and a rigidbody
- To make an object that can be grabbed/pulled by player, give it tag: **Pushable**, and a rigidbody

You're pretty much ready to go at this point. Functionality explanations and variable descriptions are commented in each script. Each script should auto-setup, and notify you of what it did. Check the demo scenes to see how things fit together.

3. Nuanced Objects

Setting up Moving Platform or Hazard:

1. Add the "MoveToPoints" script.
2. Create empty game objects, and parent them to the platform
3. Give them the tag: 'EditorOnly' (blue sphere gizmos should show now)
4. Name them in order (ie: wp1, wp2, wp3)
5. If this is a moving platform for the player to stay on, give it the tag "MovingPlatform", and in 'PlayerMove' script, tweak the 'Moving Platform Friction' variable to your liking.

Player 'FloorChecks'

Floorchecks is an object that contains empty objects, each one will cast a ray downward, half the height of the collider of the player, to see if they are standing on the ground or not.

So to set this up for your own character: create a floorChecks empty gameObject, parent some more empty gameObjects to it, and arrange them where you would like to check for the floor. Make sure they are exactly mid-way up the collider of the player, or slightly lower (else the raycast will not reach the ground). Now assign the 'FloorChecks' object to the variable: floorChecks in the player move script.

Why use this approach? It's visually friendly and easily customisable for any collider shape.

4. Script Guide

Camera Follow

How to use

Attach to your main camera

Variables

Target: the object the camera will follow and look at

Target Offset: how far away from the target should the camera be

Lock Rotation: the camera will 'lock' to this offset rotation, for example if you have an offset of -7 in the Z axis, the camera will follow behind the player

Follow Speed: how fast should camera follow object

Input Rotation Speed: when you press the "CamHorizontal" Input buttons or move the mouse, how fast should the camera rotate around the target

Mouse Freelook: when true the mouse can control the camera instead of keyboard controls, the sensitivity is adjusted by the "input rotation speed" (make sure you have 'follow speed' fairly fast, ie: 10)

Rotate Damping: how fast should camera rotate to look at the target (set to 0 when mouse freelook is on)

Water Filter: an object in front of the camera lens whose renderer gets toggled when the camera enters a water zone (is underwater)

Avoid Clipping Tags: a list of tags, if an object with one of these tags obstructs the camera it will zoom in to avoid clipping the object. (Good to put large objects in here like: houses, walls.. but not small objects such as enemies and coins)

What this script does:

Smoothly follows and look at the target. Attempts to avoid wall clipping by moving closer to the target when the intended position is obstructed by one of the objects in your 'avoid clipping tags' list. Toggles a filter when it enters water zones, to give the illusion that we are underwater. Rotates around the target when CamHorizontal input is pressed.

Extra Info:

In order to smoothly follow the target, but also rotate around it works via creating an empty object upon scene start. This empty object follows the target, rotates around target and acts as a "where the camera should be" object. Then the camera smoothly interpolates its position to this object, whilst smoothly interpolating its rotation to face the target at all times.

Character Motor

Variables

Sidescroller: sets the object up, to only move in the X/Y axis (for 2D platformers)

What this script does:

This script is attached to rigidbodies who will be moving around, like enemies or the player. It's a utility script, which means it holds methods for other scripts to make use of, but will not do much on its own. This script contains the public methods:

MoveTo(Vector3 destination, float acceleration, float stopDistance, bool ignoreY)*

RotateToVelocity(float turnSpeed, bool ignoreY)

RotateToDirection(Vector3 lookDir, float turnSpeed, bool ignoreY)

ManageSpeed(float deceleration, float maxSpeed, bool ignoreY)

*returns a Boolean: whether the object has arrived at the destination or not

Checkpoint

How to use

Attach to a checkpoint object

Variables

Active Color: the object material will be tinted this color when it is the active checkpoint

Active Color Opacity: the opacity of the material, when it is the active color

What this script does:

Sets the respawn position for the players health script, when the player enters its trigger.

Coin

How to use

Attach to any collectible object

Variables

Collect Sound: sound to play when collectible is collected

Rotation: the idle rotation

Rotation Gain: the object will increase torque by this amount when the player is within its bounds

Start Speed: when the player enters the collectibles bounds, this is how fast it starts to move toward the player like a magnet

Speed Gain: when moving toward the player, this is how fast the object gains speed

What this script does:

Senses when the player enters its “bounds” trigger and accelerates toward the player. When the collectible hits the player, it is collected, plays a sound and updates the gui.

Extra Info:

This is one of the many objects which use a child trigger object (bounds), with the ‘TriggerParent’ script attached. Trigger Parent is a utility script which checks for collisions and alerts the parent (the actual coin).

If you want a traditional collectible which does not attract to the player, attach bounds which are the same radius as the object, or set the ‘start speed’ + ‘speed gain’ to 0.

Deal Damage

What this script does:

This script is attached to any object which will be attacking other objects, like the player, enemy, or hazards. It's a utility script, which means it holds methods for other scripts to make use of, but will not do much on its own. This script contains the public methods:

Attack(GameObject victim, int dmg, float pushHeight, float pushForce)

Destroy Object

How to use

Attach to objects which you want to destroy on load

Variables

Destroy Sound: plays this sound when the object is destroyed

Delay: the delay before the object is destroyed

Destroy Children: when the object is destroyed, do we destroy the child objects as well, or detach them

Push Child Amount: pushes any child rigidbody objects away from the centre of the parent at this force (look at the broken box prefab to see how this could be useful)

Extra Info:

Use this to keep your hierarchy organised, by having parent objects as "folders", which destroy themselves and detach children on load. Use this for 'one shot' scripts which are spawned, run once, then destroyed immediately. Use this on particle effects to destroy them after a single play (look at smoke puff prefab)… this script has many simple uses.

Enemy AI

How to use

Attach to an enemy object

Variables

Acceleration: how quickly the enemy gets up to max speed

Deceleration: how slow the enemy rolls to a stop

Rotate Speed: how fast the enemy smooth rotates toward its velocity

Speed Limit: enemies move speed

Bounce Force: force applied to the player when they land on the enemy (note: this kit has consecutive jumping in, so if the player bounces on several enemies without touching the ground, each bounce will be slightly higher. You can use this to set up secret areas..)

Bounce Sound: sound to play when player lands on this enemy

Push Force: when an object is attacked by this enemy, how hard to push it away from the enemy in the X/Z axis

Push Height: when something is attacked by this enemy, how hard to push the object in the Y direction

Attack Damage: health points to remove from an object attacked by this enemy

Chase: should this enemy chase objects that enter its view?

Ignore Y: ignores movement and rotation in vertical axis. Would be true for a ground enemy, but maybe false for a flying enemy like a bird.

Chase Stop Distance: when chasing an object, how close to get to it

Sight Bounds: a trigger that represents the enemies sight. This needs the 'TriggerParent' script attached to it. You can use the 'Tags to Check' array on the 'TriggerParent' script to filter what objects the enemy will chase.

Attack Bounds: a trigger that represents the enemies attack area. This needs the 'TriggerParent' script attached to it. You can use the 'Tags to Check' array on the 'TriggerParent' script to filter what objects the enemy will attack.

Move To Points Script: if you've attached a 'Move To Points' script to this enemy, drag that component here.

What this script does:

Chases and attacks objects. Faces direction of movement. Manages the rigidbodies speed. Bounces the player when they land on this enemy, by communicating with the 'PlayerMove' script.

Goal

How to use

Attach to the end goal of the level.

Variables

Lift: objects within the trigger will be lifted up by this amount, giving it a magical victory effect

Load Delay: how long you need to stay within the goal, for it to load the next level

Next Level Index: index of the next level to load

What this script does:

A simple end to a level. It lifts the object and after a while loads the next level. In the demo scene, I used them as air lifts by setting the load delay to a ridiculous number that would never be reached. You could just as easily use the water zone script for air lifts.

GUI Manager

How to use

Attach to any object in the scene

Variables

Gui Skin: the gui skin to use when displaying information on the screen

What this script does:

Displays collectible and health information.

Hazard

How to use

Attach to anything which will push or damage objects. (lava, spikes, bouncepad)

Variables

Push Force: when an object collides with the hazard, how hard to push it away from the centre of the hazard in the X/Z axis

Push Height: when something collides with this hazard, how hard to push the object in the Y direction

Damage: damage to deal to any object that touches this hazard (which has a health script attached..)

Trigger Enter: attack objects that enter the trigger, or child trigger collider?

Collision Enter: attack objects that directly collide with the hazard? (these two variables are just to choose whether you want to look for trigger or direct collisions… different hazards will need different types of collision checks)

Effected Tags: which objects should be effected by this hazard?

Hit Sound: sound to play when an object is effected by this hazard

What this script does:

Pushes and attacks objects that collide with it, or its trigger.

Extra Info:

You can set damage to 0 if you just want a hazard to push objects, for example a bounce pad (see trampoline in the demo scene). You can set damage to a negative value for it to heal objects (which have the health script attached to them).

Health

How to use

Attach to anything which will be damaged, or have impact sounds (breakable objects, player, enemies..)

Variables

Impact Sound: sound to play when this object hits another with force

Hurt Sound: sound to play when this object takes damage

Dead Sound: sound to play when this object is destroyed

Current Health: health of the object

Take Impact Damage: should this object take damage from impacts? (falling hard on ground, being hit by thrown objects..)

Only Rigidbody Impact: if the above is true. Set this to true for it to only take damage when being hit by another rigidbody which is moving at speed (for example, you don't want enemies to take fall damage, but want them to be hurt when a box is thrown at them)

Respawn: should this item respawn when dead? (respawn point is its starting position, can be useful if you want to respawn puzzle objects, like pushable crates, which might fall off the level)

Impact Filter Tag: DON'T take impact damage from any object in this list. (in the demo scene player is filtered, else you might run fast into an enemy and hurt them)

Hit Flash Delay: how long each flash lasts when this object is damaged and starts flashing

Flash Duration: how long this object flashes after being damaged. It cannot be damaged if it is currently flashing.

Hit Flash Color: color to tint the object when it gets damaged

Flash Object: which object flashes when this object is damaged? Used for the player, where we want the child model to flash, and not the collision box. If left blank, it will just flash the object the script is attached to.

Spawn On Death: array of objects to spawn upon the objects death (! Objects all spawn at the centre of the object, so don't have multiple objects with colliders or they might clip each other)

What this script does:

Flashes the object when it is damaged, providing a small ‘invincibility window’ where it cannot be hurt immediately afterwards. Handles the death, respawning of an object and what happens when the item dies. Handles impact sounds, and impact damage.

Extra Info:

In the demo scene this is attached to some objects which won’t die usually (the metal crates), just so they can have impact sounds, and respawn if they fall off the level; there is a trigger hazard below the level which deals massive damage to anything with a health script.

Spawn on Death is a VERY useful variable, with a lot of uses. You could break a crate and have it drop a key. You could have an enemy which when killed, spawns a smaller enemy. You could have a crate of nitro take impact damage from the player, and spawn an explosion (crash bandicoot :P). You could have an empty gameObject spawn with a script that executes something, and then immediately kills itself via the ‘Destroy Object’ script, for example if you want to kill a boss and have it trigger a cutscene.

Move To Points

How to use

Attach to moving objects like platforms or hazards. (! NOT characters, those use a character motor. Objects with ‘MoveToPoints’ script attached will be kinematic rigidbodies, so they effect other rigidbodies but cannot be moved themselves, except by this script.)

NOTE: this script can be attached to enemies for patrol routes! If you attach it to an enemy, be sure to assign it to the ‘moveToPointsScript’ variable for ‘Enemy AI’ script

Variables

Speed: how fast the object moves (if the script is on an enemy, this is redundant)

Delay: how long the object stops at each waypoint

Movement Type: play once means just that, loop means it keeps looping around the points and pingpong means it moves back n forth along the points.

What this script does:

Moves an object along a series of waypoints (see chapter 3. Nuanced Objects). These waypoints should be child objects with ‘Waypoint’ tag, and numbered in order.

Extra Info:

You can get loads of gameplay out of this one script! The moving objects will effect any rigidbody (enemies, player, pickups..) realistically with inherent velocity.. you could place them underwater, you could give them the 'Pushable' tag so the player can hold onto them, you can create a conveyor belt/escalator system for moving objects…. lots of fun to be had here :]

For the player to stay on a moving platform, give it the 'MovingPlatform' tag, and adjust the 'Moving Platform Friction' variable in the 'PlayerMove' script.

Player Move

How to use

Attach to your player.

Variables

Sidescroller: constrains movement input to just the X/Y axis, if you're making a 2D platformer

Main Cam: the main camera, used to adjust controls relative to this camera

Floor Checks: object used to check if the player is grounded (! see chapter 3: nuanced objects)

Animator: animator component to animate the player

Jump Sound: play this sound when the player jumps

Land Sound: play this sound when the player lands on the ground

Accel: how fast the player gets up to max speed when on the ground

Air Accel: how fast the player gets up to max speed when in the air

Decel: how fast the player slows down

Air Decel: how fast the player slows down whilst in the air

Rotate Speed: how fast the player rotates toward its move direction

Air Rotate Speed: how fast the player rotates in the air toward its move direction

Max Speed: movement speed of the object

Slope Limit: the player will not be able to jump on slopes steeper than this angle, and the slide amount will be applied

Slide Amount: when standing on a slope steeper than the slope limit, the player will be pushed downwards by this amount (you could create fun slides with these 2 variables!)

Moving Platform Friction: adjust this value to keep the player on moving platforms. (..complex variable factors like: mass, gravity, and drag meant that to get real inherent velocity on moving platforms, this was the simplest way)

Jump Force: force applied to player on their normal jump

Second Jump Force: force applied to the player when they land a jump, and immediately jump again

Third Jump Force: force applied to the player when they land their 2nd consecutive jump and immediately jump again

Jump Delay: time after the player has landed on the ground, that they can press jump and have it perform the 2nd or 3rd jump force.

Jump Leniancy: if you press the jump button whilst still in the air, this is how long before hitting the ground, that the jump will still execute when you land. Its subtle, but makes for more responsive jumping

What this script does:

Moves and rotates the player via input. Animates the character. Stops the player sliding down subtle slopes, slides the player down steep slopes. Handles jumping. Checks if the player is grounded. Checks when the player lands on an enemy. Moves the player when they are standing on a 'Moving Platform' tagged object.

Throwing

How to use

Attach to player to allow them to pickup/throw 'Pickup' tagged objects, and allow them to grab/pull 'Pushable' tagged objects.

Variables

Pick Up Sound: sound to play when the player picks up a ‘Pickup’ tagged object

Throw Sound: sound to play when the player throws a ‘Pickup’ tagged object

Grab Box: a trigger box, which needs to be colliding with a ‘Pickup’ or ‘Pushable’ object before they can be grabbed/picked up. This of this as the players arm reach.

Gap: how high above the players head they hold objects

Throw Force: the force applied to objects when thrown

Rotate to Block Speed: when grabbing onto a ‘Pushable’ object, this is how fast the player rotates to face the object

Check Radius: when trying to pickup an object, a sphereCheck is applied above the player, to see that nothing is blocking the pickup. This is the radius size of that sphere check. (see chapter 2: things to know. If you are having problems picking up ‘Pickup’ tagged objects, select the player in scene view and gizmo spheres will be drawn when you attempt to lift objects.. it is possible the sphereCheck is colliding with the player, or a trigger and preventing the lift.)

Weight Change: when you carry an object, its weight can be reduced. So an object can appear to be heavy, but you can carry it with ease and still jump. 1 will mean no change, 0.3 will mean the object is 30% as heavy as normal whilst carried.

Holding Break Force: the force it takes to break your hold on a ‘Pushable’ object (having this too low will mean you won’t be able to pull the object much without letting go, having it too high will mean you get into weird situations, like a heavy block being off the edge of a platform but the player is still holding it)

Holding Break Torque: same as above but for the torque force (rotation)

Animator: the player animator (to animate its arms)

Arms Animation Layer: layer index of the arms in the players animator. This is used because the arms layer is blended when lifting/throwing objects. (see the demo character)

What this script does:

Checks if the player can grab or lift an object. Grabs or lifts object by attaching a physics joint between the two, when letting go or throwing the object, the physics joint is broken and force is applied to the object. It also adjusts the weight of carried objects so they do not weigh the player down too much if you still want full mobility.

Extra Info:

A public gameObject variable ‘heldObj’ will be set to whatever object the player is grabbing/carrying. You can use this variable in object scripts, for example if you want to know when a bomb is picked up so you can start it’s countdown, or a propeller which starts to lift upward when it is picked up. LOTS of gameplay and puzzle potential here!

Keep in mind that ANYTHING with a rigidbody and ‘Pickup’ or ‘Pushable’ tag applied can be lifted / grabbed. Because this is completely dynamic via physX joints, you can apply forces to objects and this will affect the player!

You can apply scripts to these objects for example you could have a cattle prod object (with rigidbody) tagged ‘Pickup’ which has a ‘Hazard’ script attached, and set to only effect enemies. Now the player can pick up this object and use it to shock enemies.

Water

How to use

Attach to a trigger box, which effects the movement of rigidbodies (mud, water, rivers, air lift)

Variables

Splash Sound: sound to play when an object enters the trigger

Force: acceleration applied to objects in this trigger

Effect Player Drag: should the players drag and angular drag values be effected whilst in this trigger? (see below)

Resistance: changes the drag of rigidbodies whilst they are in this trigger

Angular Resistance: changes the angular drag of rigidbodies whilst they are in this trigger

What this script does:

Adds force to rigidbodies, effects rigidbodies drag values, plays sounds when objects enter the trigger.

Trigger Parent

What this script does:

This is a utility script which holds information about collisions within a specific trigger, so another script can make use of that information. You can also filter which objects can collide with this trigger. It has the public variables:

Collided (a Boolean, true when an object enters the trigger)

Colliding (a Boolean true when an object stays inside the trigger)

hitObject (the gameObject which is currently inside the trigger)

5. 2D Platformer Setup

How to set your game up for 2D

Camera Follow script: make the offset a negative in the z axis, make input rotation speed 0.

(for a 'true 2D' platformer: set the camera to orthographic, with a size of around 10, make rotation damping 0 and followSpeed a very high number like 10,000)

Character Motor script: make sure the 'sidescroller' variable is set to true

Player Move script: make sure the 'sidescroller' variable is set to true, you will probably want the 'rotate speed' and 'air rotate speed' variables to be quite high

Interactive Objects: find their rigidbody component > constraints > check 'freeze Z position' (if you want objects to only rotate in the 2D angle, make sure to freeze rotation in X/Y as well)

(see the 2D demo scene for more information)

6. Replacing Character Art

How characters work:

1. Characters have a main collision box which handles all scripts and movement (for the player this has no renderer, so it is invisible)
2. A model is parented to this collision box
3. The model has an animator controller, and animations SPECIFIC to its model *(if you replace the player model, you need your own rig and animations to go with it!)*
4. *Scripts on the character pass movement information to the animator controller which uses this data to transitions between animation states*

Replacing the Player Character:

1. Find the "Player Model" parented to the "Player" and replace it with your own.
2. Your animations should be specific to your own player/rig!
3. Click on each state within the player animator controller, and in the field for "motion" select the new animation
4. For your players "arms layer" animations, make sure they have the relevant transform mask (to see this on the current player: select the player model, navigate to animations, click on an 'arms' animation, see at the bottom that the 'transform mask' is only applying animation on the arm bones, thus making it blend able in mechanim!)
5. Ensure that your player model is assigned to the 'animator controller' variables in the 'Player Move' and 'Throwing' scripts
6. Ensure the 'flash object' variable for the 'Health' script, is assigned to the player model

Adding Enemy Art/Animations:

1. Disable the enemy renderer component to make its collision box invisible
2. Parent your enemy model to the collision box
3. Assign the 'Enemy Controller' (in the misc folder) as the animator controller for this model

4. Select each animation state within this controller and assign the “motion” to your animations
5. Ensure the ‘animator controller’ variable is assigned to the enemy model, for the ‘Enemy AI’ script
6. Ensure the ‘flash object’ variable for the ‘Health’ script, is assigned to the enemy model