PolyTie Manual

Table of Contents

What to expect

The following pages will give you a brief overview of the features of the PolyTie 2D collider creation tool and explain how to use them. PolyTie will help you speed up your process in creating 2D collider objects in Unity3D. It makes drawing collider shapes very simple, similar to Vector drawing software like Illustrator. With PolyTie you can sketch out a basic platformer level in just a few minutes.

Brush Window

Clicking on Window -> PolyTie -> 2D Brushes will bring up the Brush Window. Within that window you will find quick access buttons to all the collider shapes and object types you can draw with PolyTie. You can also access those shapes under GameObject -> Create 2D Objects or by using the appropriate shortcuts.

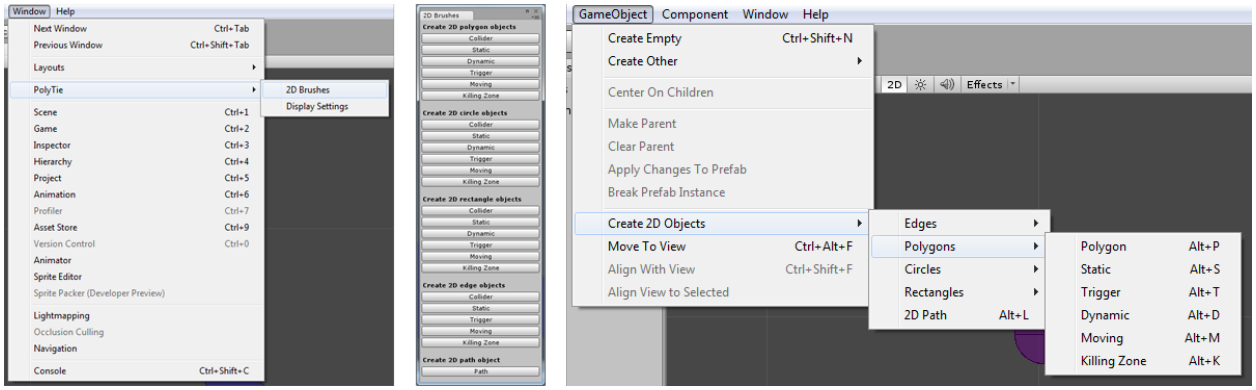

Image 1: PolyTie Brush Window, PolyTie Objects

2D Physics Debug Camera

PolyTie comes with a 2D physics debugging script, that draws all collider shapes all the time in the editor's scene view as well as in game. This is especially useful if you don't have any artwork yet for a level and simply want to test some concepts. It also allows you to select 2D collider objects within the scene view by just clicking inside of them. To add the debug view to your project, simply add the *Physics2DDebugRenderer* script to your main camera object. You can find it in the PolyTie/Scripts/Tools folder. Now all your collider objects should be drawn with filled colours. Alternatively you can select Edit -> PolyTie -> Add Debug Camera to add it automatically to the main camera in the scene.

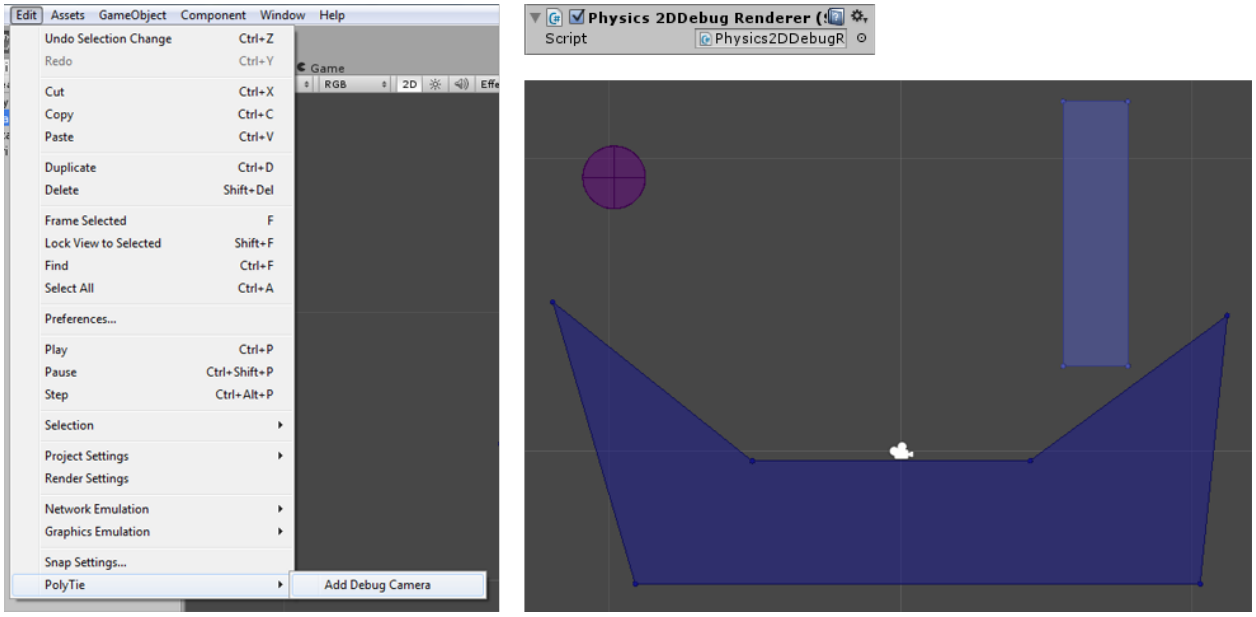

Image 2: Add Debug Camera Menu, Debug Camera Script, Debug Camera Display

Brushes

PolyTie features brushes for all the possible 2D collider shapes. You can access them by either the quick access buttons in the Brush Window, or by going to GameObject -> Create 2D Objects. The following sections will look at each collision shape in more detail and explain how to create them.

Polygon Brush

Polygons are probably the most commonly used shapes. You can start drawing by pressing the according quick access buttons or by selecting one of the objects under GameObjects -> Create 2D Objects -> Polygons. You can then start placing vertices by left clicking on the scene view. It is possible to draw concave polygons as well as convex, Unity automatically splits them up into the necessary type. To stop drawing and close off the polygon once you're satisfied simply press the right mouse button. To create a valid polygon you have to place at least three vertices before closing it off. Once you have closed off a polygon you can still manipulate it. To manipulate a vertex simply hover over it and it will turn red. Then hold the left mouse button and drag the vertex to the new position. If you hold shift while dragging, the vertex will snap to either the x- or y-Axis depending to which it is closer - similar to Photoshop. If you hold the "V" key the vertex will snap to the vertices of other collider objects. To add a new vertex to the polygon simply double click on an edge of the polygon. To delete one of the vertices, right click on the vertex you want to delete and select delete from the drop down menu.

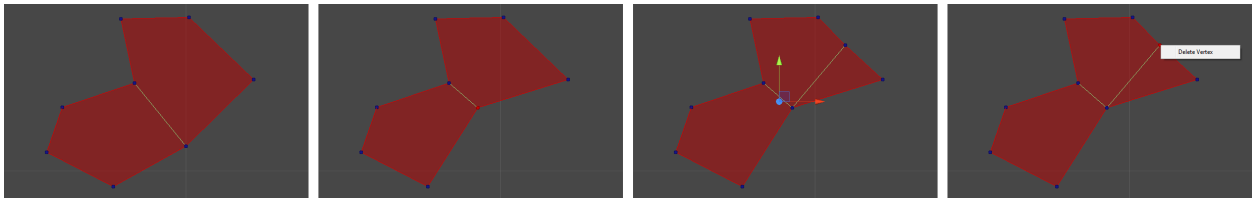

Image 3: Create Polygon, Move Vertes, Add Vertex, Delete Vertex

Circle Brush

The object types using this brush are accessible by its according quick access buttons or selecting on of the objects under GameObjects -> Create 2D Objects -> Circles. You draw a circle by clicking in the scene view and then dragging it out. The first click determines the center of the circle. Dragging the mouse will change the radius. Once you let go of the left mouse button, the circle is closed off. You can change the radius of the circle by hovering over its edge. The edge will turn red and you can now click and drag to change the circle's radius.

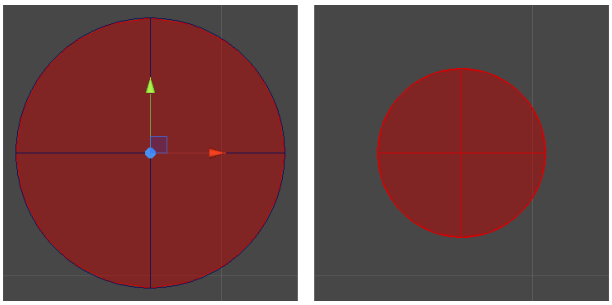

Image 4: Create Circle, Change Radius

Rectangle Brush

You can draw rectangles by clicking on the according quick access buttons or by selecting one of the objects under GameObjects -> Create 2D Objects -> Rectangles. To draw a rectangle click and drag within the scene view. The first click determines the first corner of the rectangle. Dragging the mouse will change the size of the rectangle. To change the rectangle shape you can hover over one of the corners. The corner will turn red and you then can click and drag to change its position. If you hold the "V" key the corner will snap to the vertices of other collider objects. You can also hover over an edge. The edge will turn red and you then can click and drag it to move it perpendicular to itself.

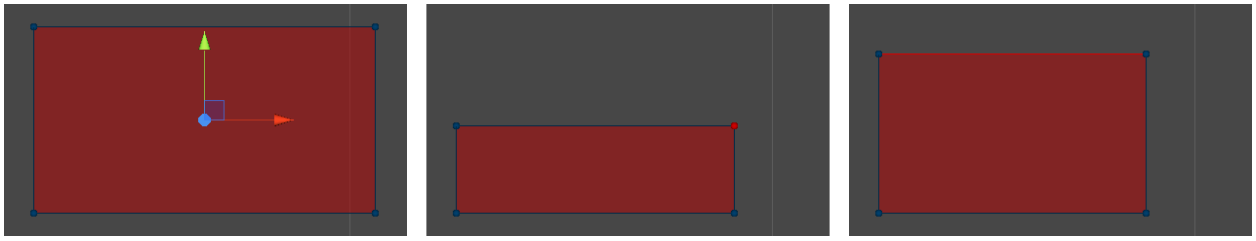

Image 5: Create Rectangle, Change Corner, Change Edge

Edge Brush

To create an edge click on the according quick access buttons or go under GameObjects -> Create 2D Objects -> Edges. This will place the first vertex at the origin of the scene view. This is due to how Unity handles edge colliders. Once you press the left mouse buttons you add vertices to the edge. As soon as you add the third vertex the first vertex at the origin is deleted. Right clicking will close off the edge. You can manipulate the edge by hovering over a vertex. It will turn red and you can then click and drag to manipulate its position. If you hold shift while dragging it will snap either to the x- or y-Axis, depending on which is closer - similar to Photoshop. If you hold the “V” key the vertex will snap to the vertices of other collider objects. To add a vertex simply double click on the edge objects. To remove a vertex right click on a vertex and click delete.

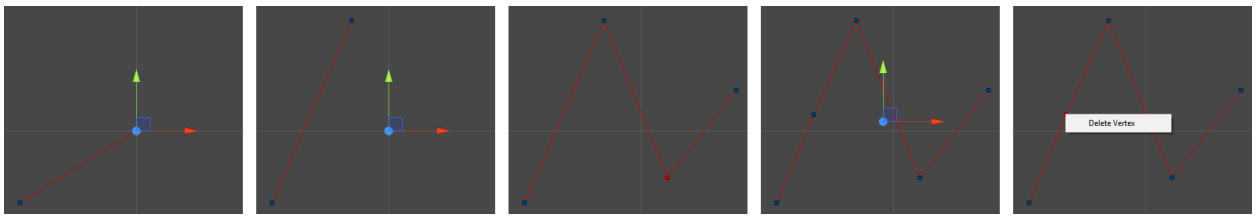

Image 6: Start Edge, Create Second Vertex, Move Vertex, Add Vertex, Delete Vertex

Path Brush

To start drawing a brush click on its quick access button, select GameObjects -> Create 2D Objects -> 2D Path or press Alt + L. Left click into the scene view to start placing vertices. To close off the path simply press the right mouse button. You can manipulate the path by hovering over a vertex. The vertex will turn red and you can click and drag to manipulate its position. If you hold shift while dragging it will snap to either the its x- or y-Axis, depending on which is closer - similar to Photoshop. If you hold the “V” key the vertex will snap to the vertices of other collider objects. To add a vertex simply double click on the path. To delete a vertex right click on a vertex and select delete.

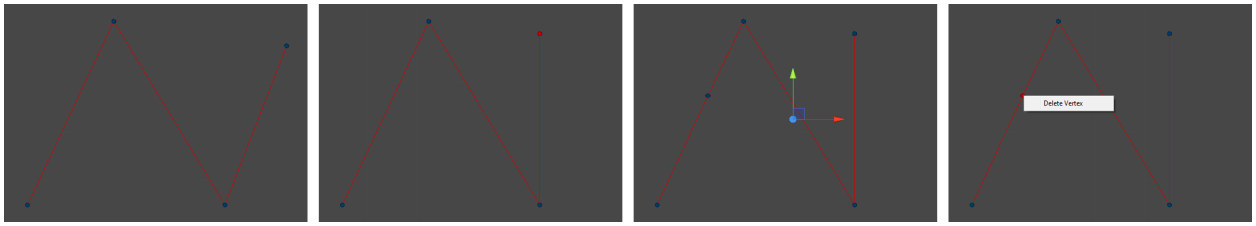

Image 7: Create Path, Move Vertex, Add Vertex, Delete Vertex

Object Types

PolyTie features the standard 2D collider and physics types that are integrated with Unity. On top of that it comes with two of it's own types, the moving object and the killing zone object. With PolyTie it is very easy to create static or dynamic 2D physics objects. If you click for instance on the static quick access button within the polygon category it will automatically add a 2D rigid body to the static collider and set its flag to kinematic. You can also quickly draw out objects that move along a path or objects that kill or respawn other objects.

Static Objects

A static object is simply a 2D collider with a 2D rigid body attached to it where the checkbox "Is Kinematic" is checked. To create for instance a static polygon simply press Alt + S. Then the polygon drawing process is started and the 2D rigid body is added and the kinematic flag is set automatically. These types of objects are usually used as floors or walls to shape the general structure of the level. They are not affected by gravity and you can't apply forces on them. They can have a velocity though. We essentially use the term static object for a kinematic object with zero velocity.

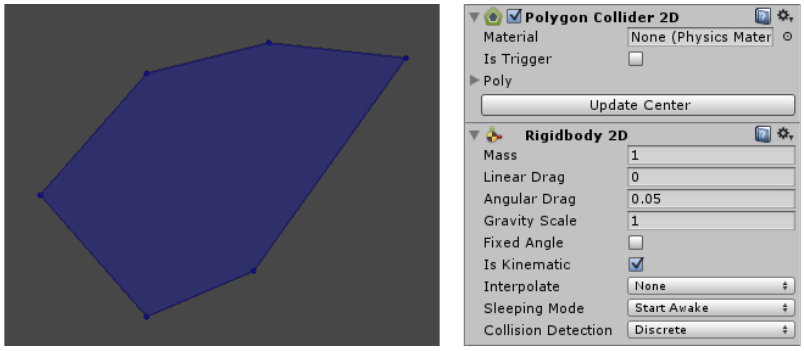

Image 8: Static Polygon, Physics Properties

Dynamic Objects

A dynamic object has a 2D rigid body attached to it and the "Is Kinematic" flag is set to false. To create for instance a dynamic polygon simply press Alt + D. Then the polygon

drawing process is started and the 2D rigid body is automatically added. Dynamic objects are usually used for things like falling rocks, bullets, crates etc… . They are by default affected by gravity and you can apply forces on them.

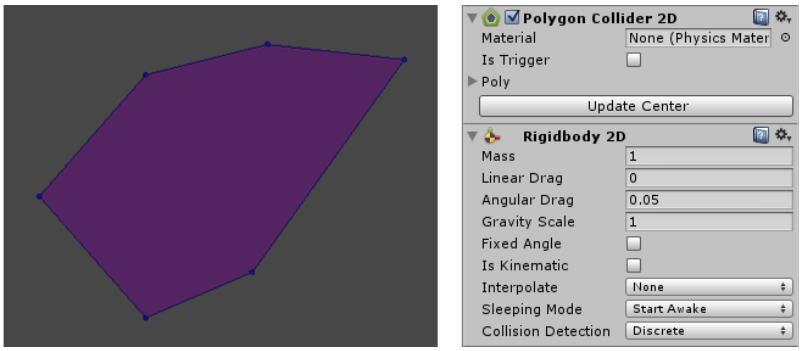

Image 9: Dynamic Polygon, Physics Properties

Trigger Objects

Trigger objects are 2D colliders with the “Is Trigger flag” set to true. To create for instance a trigger polygon simply press Alt + T. Then the polygon drawing process is started and the “Is Trigger” flag is automatically set to true. Triggers are not affected by forces and don’t block other objects. They simply fire events once other physical objects interact with them. They send a message once an object enters the trigger area, every frame an object stays in that area and once it leaves the area. To access those events you can simply attach your own script to the trigger objects and add the corresponding trigger functions, *OnTriggerEnter2D*, *OnTriggerStay2D* and *OnTriggerExit2D*. You can find more detailed information about the trigger system in the Unity documentation. PolyTie also provides a trigger controller script that is automatically added to the trigger objects once you have created them. With it you can specify a target to route those events to. These events are then triggered on the specified target as well. You can specify whether you want only the target, the object the trigger collider is on or both to receive those events. You can also specify whether it should listen to the enter, the stay or the exit events by setting the check boxes accordingly. You can also specify that the trigger is destroyed once an event is fired by setting the according check box.

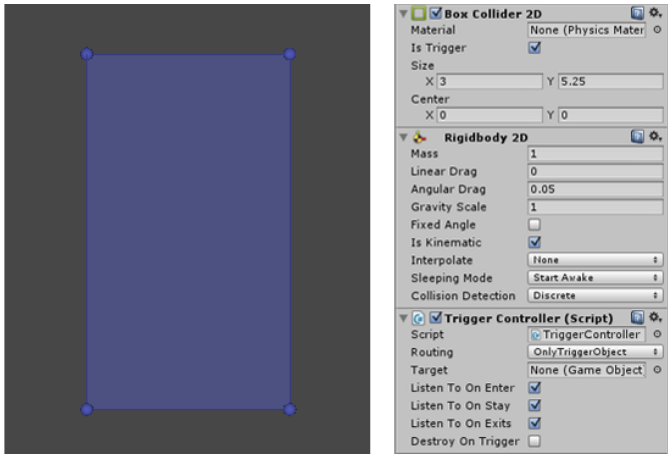

Image 10: Trigger Rectangle, Trigger Controller Properties

Moving Objects

Moving objects are kinematic objects that follow a specific path. You can create for instance a moving polygon by pressing Alt + M. This starts the polygon drawing process and it automatically adds all necessary controllers and motors for a moving object. Once you have created a moving object you have to assign a path for it to follow. You can create a path by pressing Alt + L. If you want the object to patrol along the path check the “Is Looping” box. If you want the object to start moving when the game starts you need to check the “Move On Start” box. Otherwise you need to use a trigger and rout its on enter event to the moving object. This can be used for instance to create a lift or a trap in combination with a killing zone object.

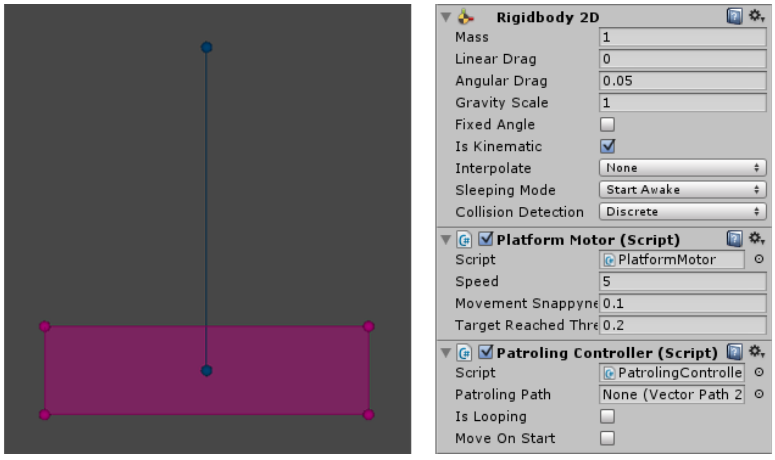

Image 11: Moving Rectangle with Path, Moving Object Scripts

Killing Zone Object

A killing zone object destroys or respawns all objects it touches. You can for instance

create a killing zone object pressing Alt + K. This will start the polygon drawing process and it automatically adds all necessary controller scripts to the object. If you don't specify a respawn point this will basically destroy every object it touches. If you assign a respawn point to it objects that touch the killing zone are first tried to be respawned by the respawn point. The object is then respawned at the specified location or is destroyed. You can find a respawn point prefab in the Prefabs folder in the PolyTie folder. The respawn controller script can be extended easily to allow for certain respawn behaviour. You might only want to respawn player characters. Other than that you can also specify affected layers. Only objects that are part of one of those layers are destroyed or reset.

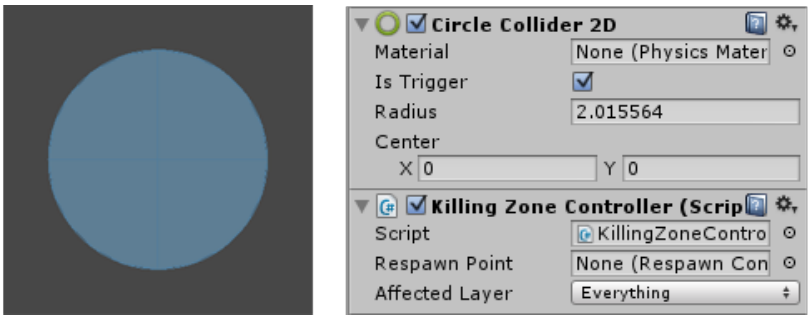

Image 12: Killing Zone Circle, Killing Zone ControllerPath Object

A path object is simply a list of vertices. Path objects are used in poly tie to specify a trajectory path for moving objects, but they can basically used for anything that requires a sequence of positions. For instance you could use a path object to specify AI waypoints or a specific sequence of collectibles that have to be collected in a specific order.

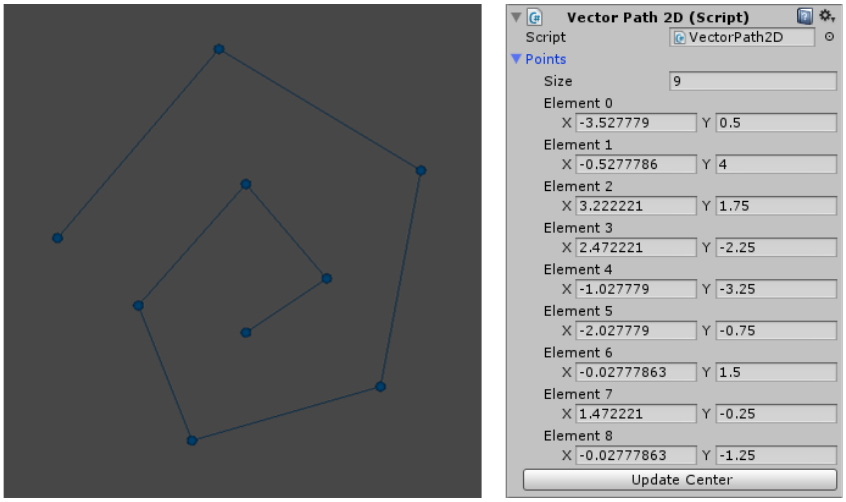

Image 13: Vector Path, Properties

Settings Window

To access the settings window go to Window -> PolyTie -> Display Settings. Within the window you can change various visual aspects of how 2D colliders are presented with PolyTie, as well as grid snapping options.

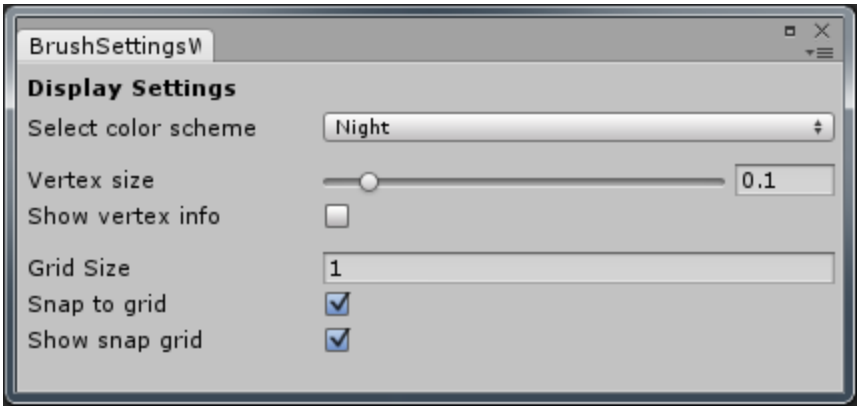

Image 14: Display Settings

Colour Scheme

Here you can change the colour pallets that are currently used to draw the collider objects. You can also create your own colour pallets. Look at the chapter Colour Pallets for more information about that.

Vertex Size

Change the size of the circles that are drawn to represent the vertices of polygons, edge, rectangle and path objects.

Show Vertex Info

Shows the position information of vertices within an object. This is useful for debugging.

Grid Size

Change the size of the grid in Unity units.

Snap to Grid

Check this box if you want to snap to the grid cells specified by the grid size.

Show Snap Grid

Draws the grid specified by the grid size in the scene view.

Colour Pallets

PolyTie uses colour pallets to draw the various collision objects. It ships with three standard colour pallets, *Dawn*, *Forest*, and *Night*. You can access them under PolyTie/Resources/ColorPallets. If you click on them you can access all the colours for the different collider types and change them to your liking. You can also create your own colour pallets by simply duplicating existing ones and renaming them or by right clicking within the project view and selecting Create -> Color Pallet.

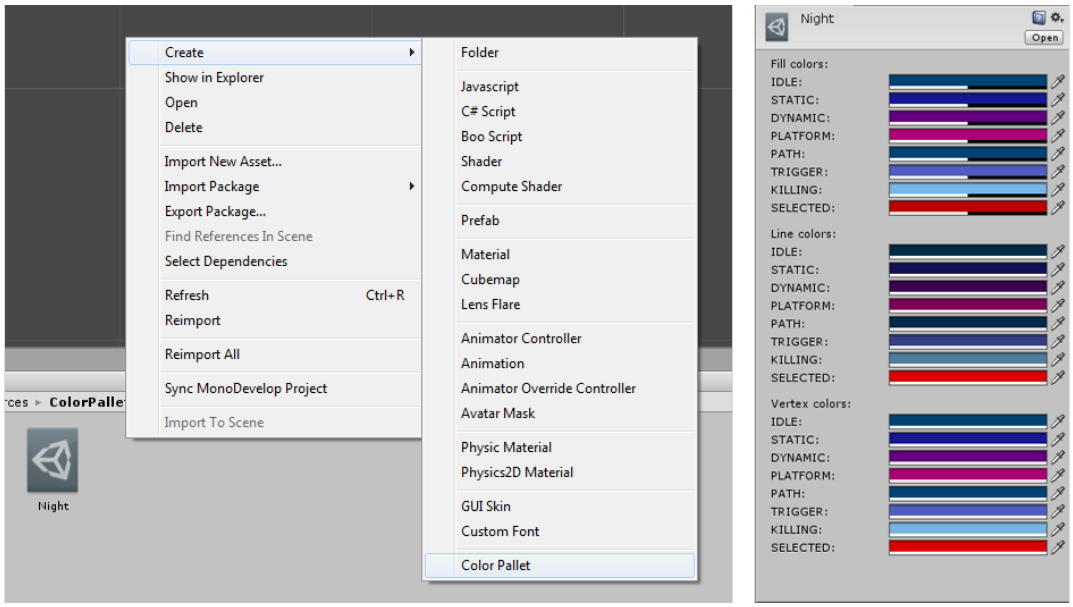

Image 15: Create Colour Pallet, Change Colours

PolyTie Importer and Exporter

This tool comes with a layout exporter and a level importer function that should simplify your workflow and speed up the asset integration process.

Level Importer

The level importer takes some sprites and meta information in a xml file and imports them into Unity according to the position information in the xml file. To start the import process simply right click on an existing xml file. The option Import to Scene should show up. Select it to start the import process. Depending on the amount of sprites you import this way, the process can take a few minutes because it also automatically sets the compression size of the imported sprites to a higher value if necessary. This only works if you perform that action on a valid xml meta file that adheres to the structure required by the PolyTie importer. To automatically produce an xml file together with the sprites you can use the

PolyTie Photoshop Exporter. The level importer tool is based on the smooth 2D workflow by Brett Bibby shown in this blog post.

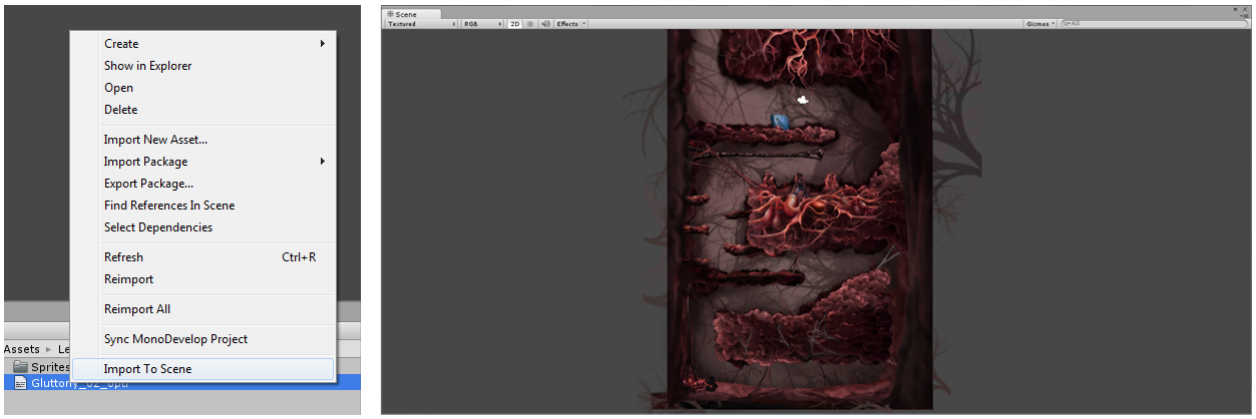

Image 16: Import To Scene Menu, Imported Scene

Layout Exporter

The layout exporter is a simple tool to save your layout as a list of png files with a xml file containing meta data. You then for instance can hand this exporter layout to an artist, who then can draw nice things around your collision shapes. To access the PolyTie layout exporter you have to drop the *LayoutExporterButton* script on one of your cameras in your scene. You can find this script in PolyTie/Scripts/Tools/LayoutExporterButton.cs. Now once you run the game a button should show up saying Export Layout. If you press that button it will start exporting the scene, which can depending on the size of your scene take a while. The exporter simply takes screenshots from your scene and automatically moves the camera along to cover all collider objects in the scene. It also produces an xml file containing the position information. This xml file can then be used for instance by the PolyTie Photoshop importer script to create a nice flat image from the tiles you exported. The xml file and the tile images can be found in the folder PolyTie/LayoutExports/{SceneName}/.

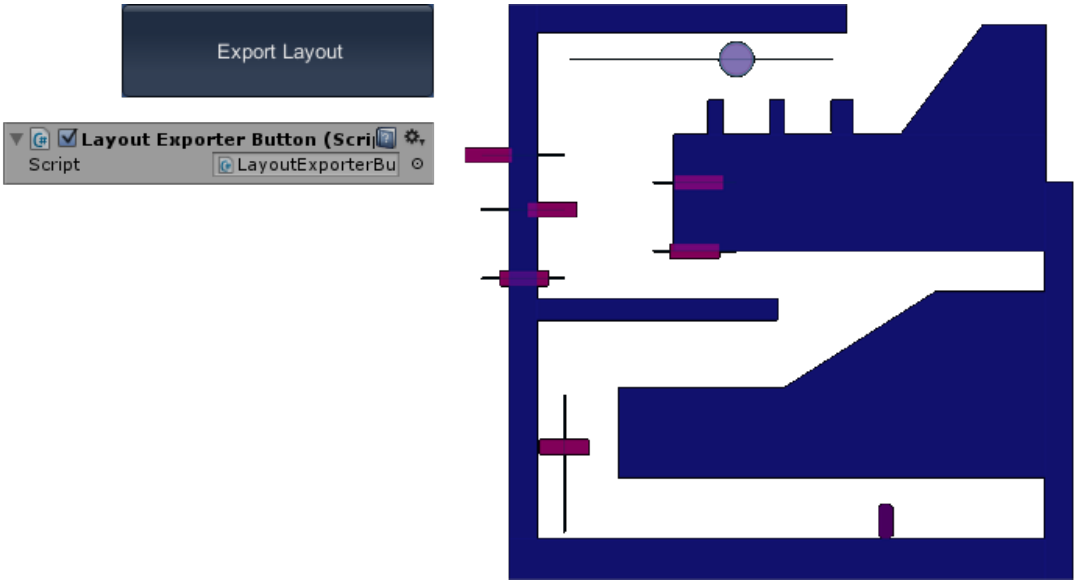

Image 17: Export Layout Button, Export Layout Script, Exported Layout

Photoshop Scripts

PolyTie comes with two Photoshop scripts that should simplify your workflow and work together with the layout export and the level import function of PolyTie.

PolyTie Photoshop Exporter

This script will automatically export all the layers of your photoshop files as png files with power of two sizes and produce an xml file containing meta information that can be read by the PolyTie level importer tool. It also automatically cuts images that are bigger than 4096 pixels into tiles so they can be imported by Unity. The script can be found in the PolyTie/PhotoshopScripts folder. To install it simply copy it to the Presets/Scripts folder within your Photoshop installation. To run it select in Photoshop File -> Scripts -> PolyTieUnityExporter. A window should pop up asking you to specify an export destination. Once you did so the script will start exporting your currently open Photoshop document. The Photoshop script follows certain conventions. All groups in the Photoshop files will be considered as Unity-Layers in the xml meta information. All layers within that group are exported as sprite objects of that Unity-Layer. All layers in Photoshop that do not reside within a group are considered annotation layers and are ignored by the script. The export process can take a few minutes depending on the size of your photoshop document. At the end you should have a xml file containing meta information and a folder with png files containing the cut and resized layers as png files. This xml file can then be used by the PolyTie Level Importer tool to bring the sprites into Unity and position them exactly as they were positioned in Photoshop.

PolyTie Photoshop Importer

This script produces one big image from the tiles and the meta information in the xml file, exported by the PolyTie Layout Exporter. You can find the script in the PolyTie/PhotoshopScripts folder. To install it simply copy it to the Prestes/Scripts folder within your Photoshop installation. To run it select Photoshop File -> Scripts -> PolyTieLayoutImporter. A dialog will pop up asking you to specify the path to a xml file. The meta information in that xml file will then be used to correctly layout the tiles. If you selected a valid xml file the import process will start. It first creates a Photoshop document with the necessary size. Then it will open all the tiles in sequence and paste them into the photoshop document created before aligning them exactly as specified in the xml file. Finally the document is flattened. Depending on the size of your layout this process can take a few minutes to complete.

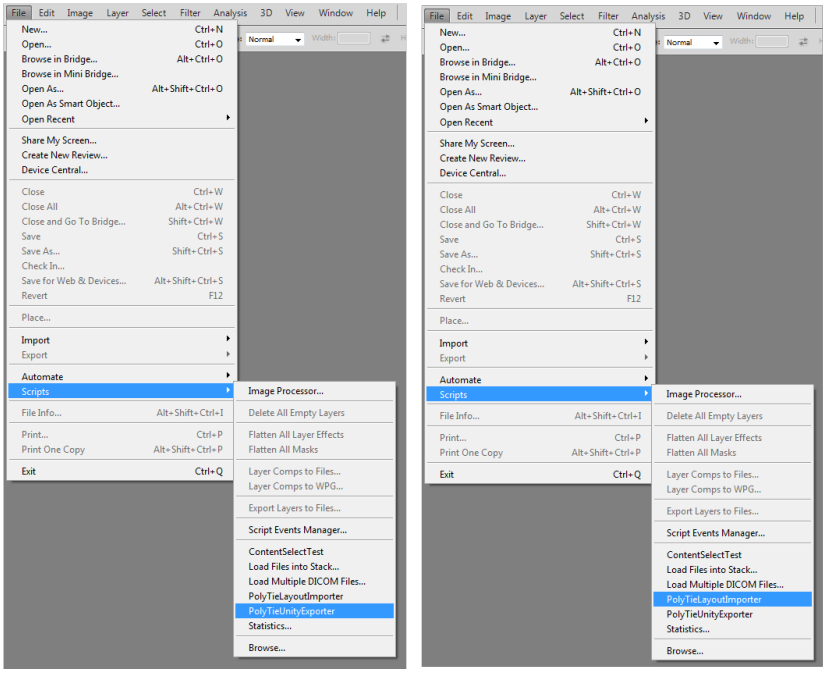

Image 18: Export from Photoshop to Unity3D, Import from Unity3D to Photoshop

Appendix

Shortcuts

Alt + P: Create new polygon collider
Alt + S: Create new static polygon
Alt + D: Create new dynamic polygon
Alt + T: Create new trigger polygon
Alt + M: Create new moving polygon
Alt + K: Create new killing zone polygon
Alt + L: Create new path
Hold Shift key: Snap to X- or Y-Axis
Hold Ctrl key: Snap to grid
Hold “V” key: Snap to vertex

Example Level Importer XML file

```
<?xml version="1.0" encoding="utf-8"?>
<Level xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Name="Example-Level-Meta-Information">
  <Layers>
    <Layer Name="Front">
      <Sprites>
        <Sprite Name="left_1-q00" Width="256" Height="512">
          <FileName>Sprites/TreeInFront-layer-0-0.png</FileName>
          <x>-722.5</x>
          <y>1305</y>
        </Sprite>
      </Sprites>
    </Layer>
    <Layer Name="Background">
      <Sprites>
        <Sprite Name="BackgroundSky-q00" Width="1024" Height="4096">
          <FileName>Sprites/BackgroundSky-layer-1-0-q00.png</FileName>
          <x>-425.5</x>
          <y>1061.5</y>
        </Sprite>
        <Sprite Name="BackgroundSky-q01" Width="1024" Height="1024">
          <FileName>Sprites/BackgroundSky-layer-1-0-q01.png</FileName>
          <x>-425.5</x>
          <y>1061.5</y>
        </Sprite>
      </Sprites>
    </Layer>
  </Layers>
</Level>
```

Example Layout Exporter XML file

```
<?xml version="1.0" encoding="utf-8"?>
<Layout xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" Name="">
   <SceneWidth>2874</SceneWidth>
   <SceneHeight>1917</SceneHeight>
   <Tile Width="1437" Height="639">
         <FileName>Tiles/-LayoutTile-0-0.png</FileName>
         <PosX>0</PosX>
         <PosY>0</PosY>
   </Tile>
   <Tile Width="1437" Height="639">
         <FileName>Tiles/-LayoutTile-0-1.png</FileName>
         <PosX>1437</PosX>
         <PosY>0</PosY>
   </Tile>
   <Tile Width="1437" Height="639">
         <FileName>Tiles/-LayoutTile-1-0.png</FileName>
         <PosX>0</PosX>
         <PosY>639</PosY>
   </Tile>
   <Tile Width="1437" Height="639">
         <FileName>Tiles/-LayoutTile-1-1.png</FileName>
         <PosX>1437</PosX>
         <PosY>639</PosY>
   </Tile>
   <Tile Width="1437" Height="639">
         <FileName>Tiles/-LayoutTile-2-0.png</FileName>
         <PosX>0</PosX>
         <PosY>1278</PosY>
   </Tile>
   <Tile Width="1437" Height="639">
         <FileName>Tiles/-LayoutTile-2-1.png</FileName>
         <PosX>1437</PosX>
         <PosY>1278</PosY>
   </Tile>
</Layout>
```