Ultimate Isometric Toolkit - User Documentation

for version 2.0

Introduction

This is the user documentation for the Ultimate Isometric Toolkit. For a quick setup see the Quick Guide

Contents

1 IsoSorting

IsoSorting is the key component of every isometric scene. It is responsible for sorting all IsoTransforms according to the given sorting strategy There are multiple ways on how to achieve proper isometric sorting, each of which makes different assumptions about your game. It is therefore crucial to choose the right *sorting strategy*. The isometric toolkit offers three sorting strategies right now.

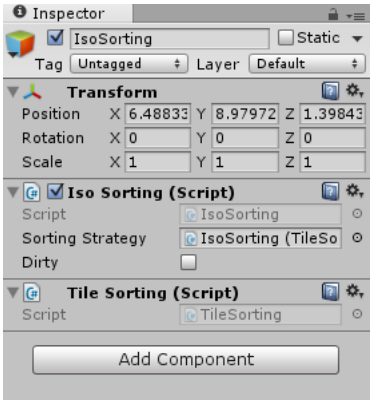

Figure 1: IsoSorting component

1.1 Heuristic Sorting

Fastest sorting strategy with smallest cpu overhead that requires all sprites to be of relatively equal size. Use for games with the maximum number of sprites that do not differ much in size. Sorting overhead can be neglected.

1.2 Tiled Sorting (pro)

Second fastest sorting strategy that requires all sprites to be tiled (sliced into 1x1 pieces). Use for games with a large number of sprites. *This is the sorting strategy you would use in most cases.*

1.3 Dependency Sorting (pro)

Worst in terms of cpu overhead, best in overall compatibility. Allows objects to be of any size. Use for games with up to a couple of hundred sprites.

Please note that perfect isometric sorting in all cases can not be achieved. All sorting strategies have edge cases in which visually correct sorting is impossible. That is not only for this toolkit but for any sorting approach. The provided sorting strategies are just the most common approaches.

2 IsoTransform

IsoTransforms (former IsoObjects) extend the regular Transform component similar to the RectTransform component. The IsoTransform component replaces the Transform component in the inspector window. Every GameObject in your Scene that has to hold information like a position or size must have an IsoTransform component attached.

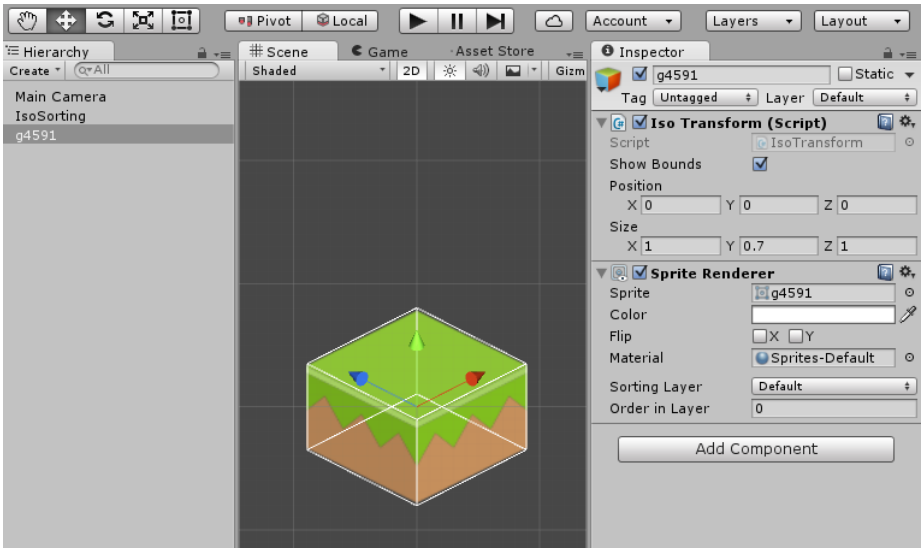

Figure 2: IsoTransform

3 IsoSnapping

Shortcut CTRL+L. The IsoSnapping tool allows you to evenly place IsoTransforms in your scene.

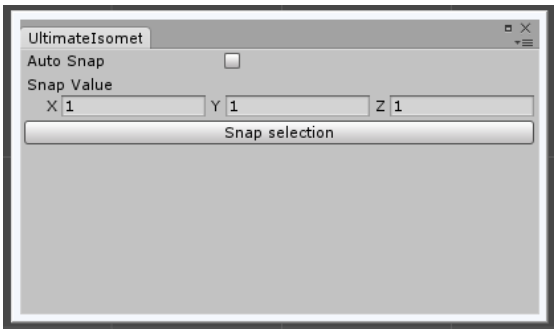

Figure 3: IsoSnapping

4 IsoController

IsoControllers are character controllers. For compatibility reasons I will desist from renaming those classes.

4.1 SimpleIsoObjectController

Basic controller that takes Arrow keys/WSAD input and translates the IsoTransform component. Does not cover collision detection/gravity/etc.

```
void Update() {
        //translate on isotransform
        _isoTransform.Translate(new Vector3(Input.GetAxis("Horizontal"),
        0, Input.GetAxis("Vertical")) * Time.deltaTime * Speed);
}
```

4.2 AbstractIsoObjectController (pro)

An abstract implementation of an advanced controller that allows objects to move from a position to a position using a custom easing function. Feel free to extend this class by writing your own derived class that makes use of such functionality.

4.3 AdvancedIsoObjectController (pro)

Extends the SimpleIsoObjectController with physics support.

5 Physics (pro)

The Ultimate Isometric Toolkit comes with physics support. This is done using hidden *ghost* objects that map any forces to their respective IsoCollider and vice versa. Both triggers and collisions are supported.

5.1 IsoRigidbody

Equivalent of the regular Rigidbody component. Allows to apply forces to this IsoTransform.

Similar to the Rigidbody component OnCollisionEnter/OnCollisionStay/OnCollisionExit and OnTriggerEnter/OnTriggerStay/OnTriggerExit will be invoked. See *SuperAwesomePhysicsScript* for exemplary use.

5.2 IsoCollider

The base class of isometric colliders. Similar to the regular unity colliders all IsoTransforms that have an IsoRigidbody or that other IsoRigidbodies should collide with need an IsoCollider attached. For ease of use you can use one of the default colliders.

5.3 IsoBoxCollider

Equivalent of the BoxCollider. Default collider that will be attached when adding the IsoRigidbody component.

5.4 IsoCapsuleCollider

Equivalent of the CapsuleCollider. Most useful collider when working with IsoRigidbodies and character controllers.

5.5 IsoSpehereCollider

Equivalent of the SphereCollider.

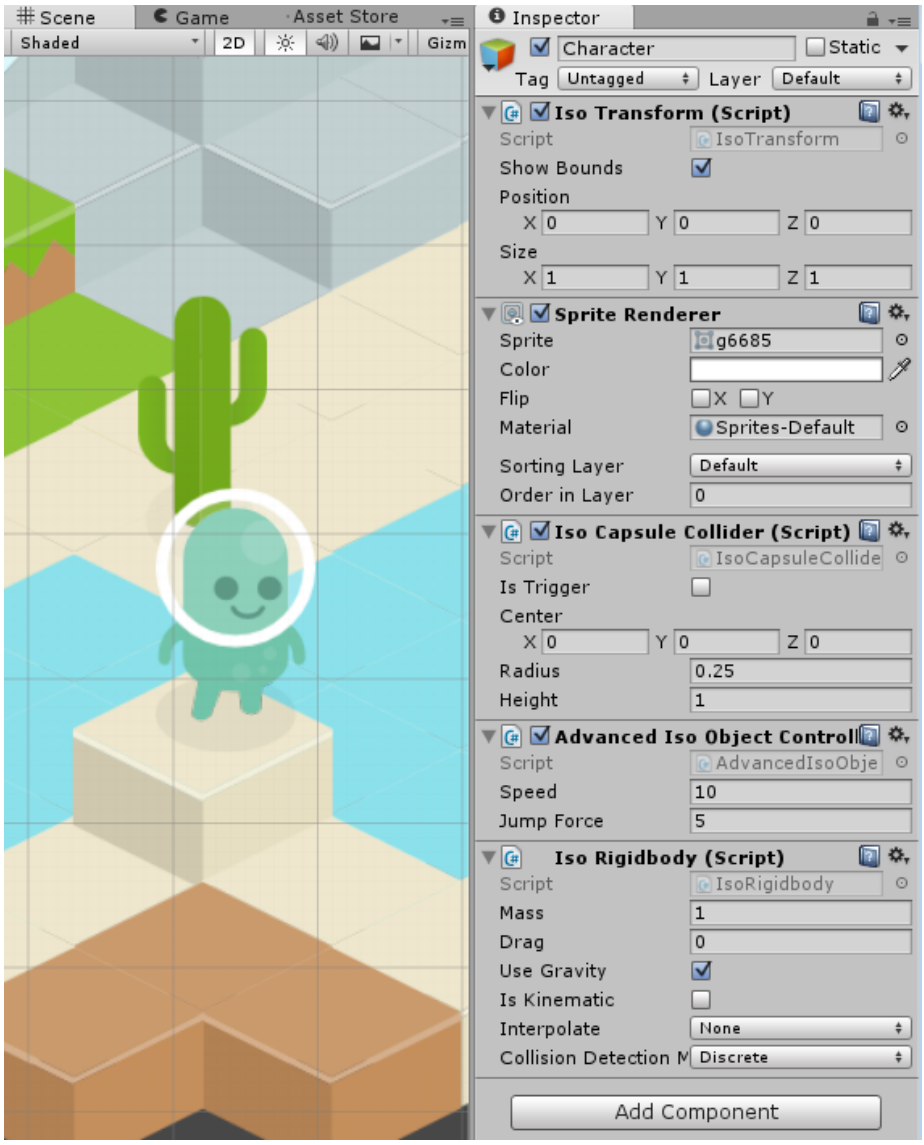

Figure 4: IsoCollider, IsoRigidbody and Controller on the character object

6 Pathfinding (pro)

After many requests the isometric toolkit now provides pathfinding support using the A* algorithm. The A* algorithm operates on a graph we therefore have to provide such a data-structure for our *AstarAgent*.

6.1 GridGraph

The GridGraph component discretizes our isometric scene into equal tiles of 1x1s. It then searches for all IsoTransforms in the scene and creates a "negative image" of it to detect empty spaces called *gaps* which are the nodes in our graph. It then figures out transitions between those gaps which make up the entire graph that our *AstarAgent* can walk on.

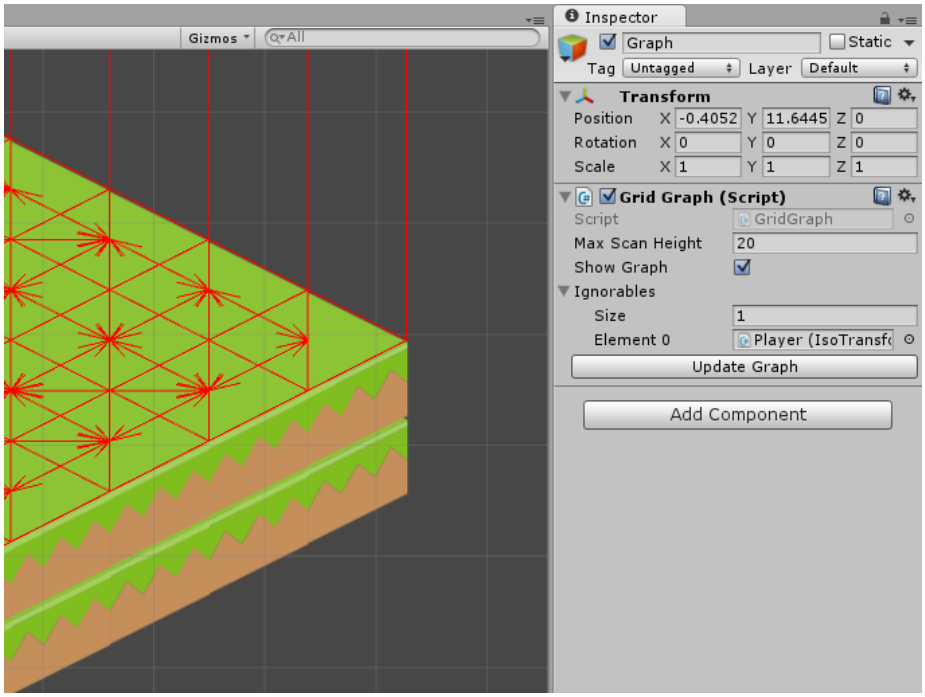

Figure 5: GridGraph

Max Scan Height the maximum height of the resulting graph

Show graph when active shows the gaps and its transitions to other gaps

Ignorables IsoTransforms that should be ignored when constructing the graph such as characters, small objects, etc.

Update Graph Reconstructs the graph, please note that this is a resource intensive operation and must not be called per frame.

6.2 AstarAgent

The AstarAgent component must be attached to all IsoTransforms that should walk on our path. Path finding is done on a different thread for maximum performance.

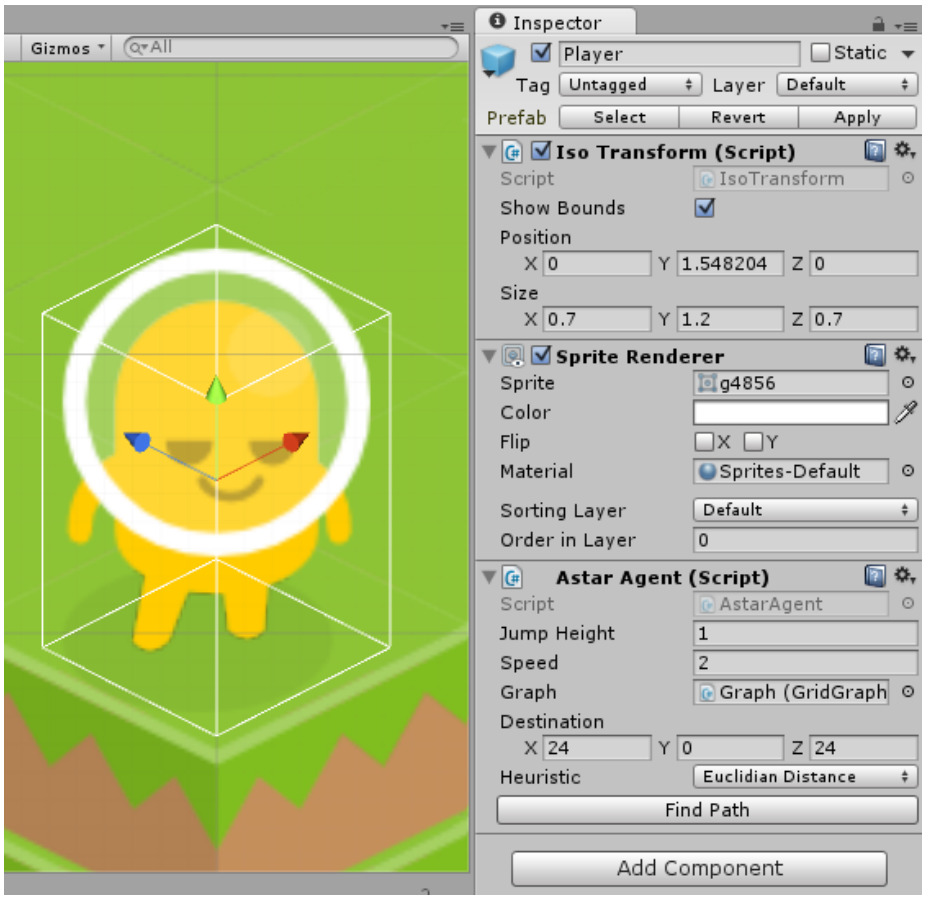

Figure 6: AstarAgent

Jump Height the maximum vertical distance the agent will travel from one node to another

Speed The agent's speed in units per second

Graph The graph the agent will operate on

Destination The destination to walk to

Heuristic the agent's heuristic when calculating a path

7 LevelGenerator (pro)

The isometric toolkit has a level generator that creates Minecraft-esque levels using a 3d perlin-noise function. It is therefore pseudo random and will generate the same

level given the same input.

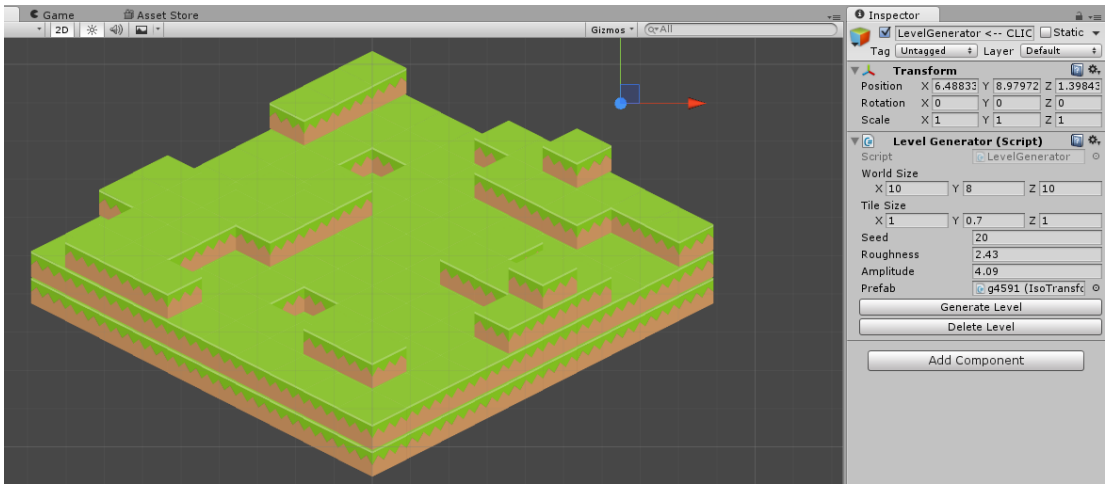

Figure 7: LevelGenerator

World Size the generated level size

Tile Size The size per tile in x,y,z should be equivalent to the prefab's size

Seed Seed value to generate a different level under the same values for roughness and amplitude

Roughness Terrain roughness

Amplitude Terrain amplitude

Prefab The prefab used to generate the level with.

8 Isometric

Isometric helper class that offers utility functions.