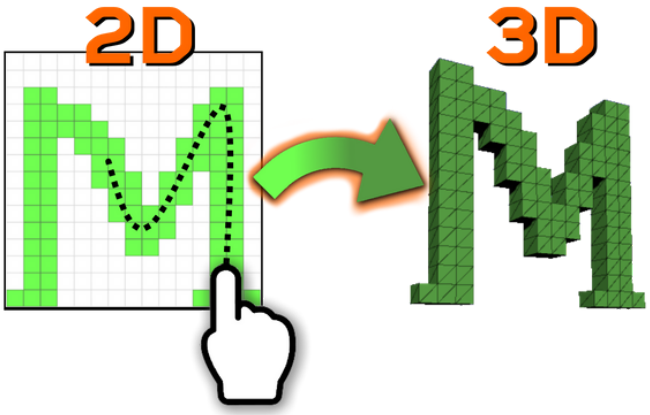
# Mesh Pencil

## asset documentation

**CONTENT** **PAGE**





# Introduction

## About

Mesh pencil is an asset for unity which converts your 2D input by drawing to 3D mesh.
The asset works in realtime play mode and not able to use it in the editor scene.
Is not require special libraries and packages for installing

## Possibilities

### Configure draw canvas area

Easy setup a pixel canvas with custom sizes by configuring pixels quantity in a row and columns amount.
Control a single pixel dimension, sprite and colors for your style and design requirements

### Common input commands

The input controller has base functions for painting, erasing, brush size control.
You could customize the cursor draw/erase mode sprites.
Supports input from the mouse for pc and by touches + UI elements from mobile devices.

### Creating the collider

Rigid body component is not supports non-convex mesh collider.
For physics features , the asset able to create a dynamic “complex box collider”.

## Save/Load

There are methods for save/load drawn data into a file which will be placed in the persistent data folder.
It helps load lightweight data files in realtime.
Especially useful for multiplayer mechanics

## Supported platforms

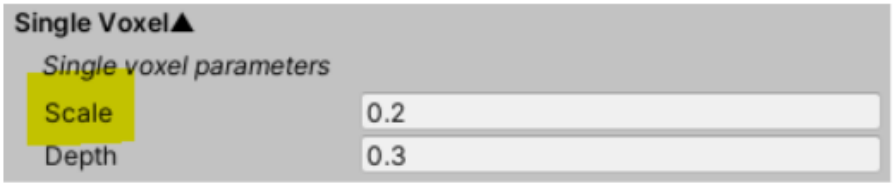
The asset supports most popular platforms :

- PC (WINDOWS, MAC, LINUX)
- MOBILE(ANDROID, IOS)
- CONSOLE (XBOX, PS4)
- WEB
- VR/AR

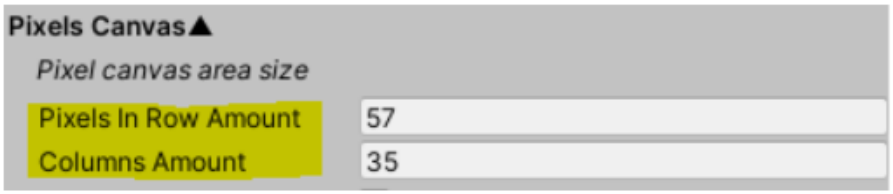
## Restrictions

- The canvas works only with X/Y oriented axis and not support X/Z or other directions
- Don`t change the MeshPencil prefab “Transform.Scale” values, it should be 1.1.1

If you need to change the canvas scale, use the controller interface for it.Increase pixel scale value in the MeshPencilController component




and change the Column/Row Amount values




## Render Camera

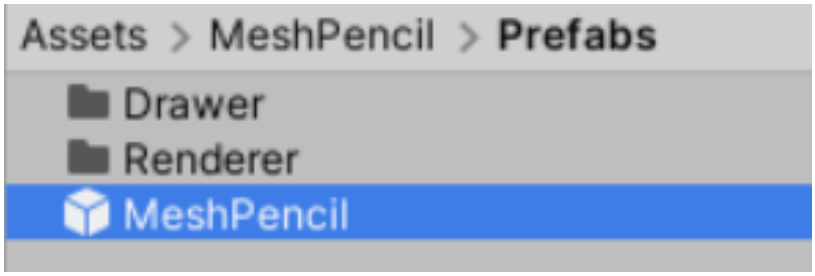
You could place the MeshPencil prefab to any position and render it from special camera created for that.

[Tutorial how to setup render camera]

# Fast setup

## Fast setup instruction

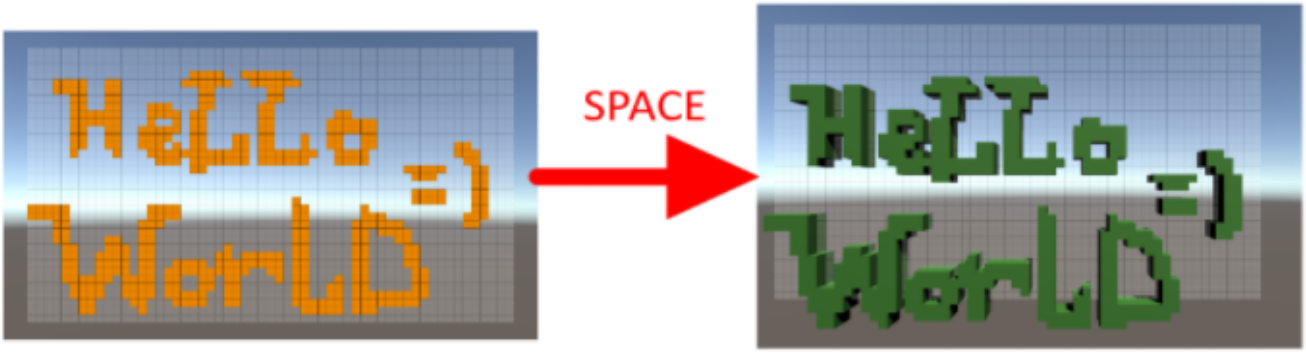
1)Import the package into your assets folder
2)Add a “MeshPencil” prefab to your scene from “*Assets/MeshPencil/Prefabs/*” folder




3)Move the prefab to needful place

**Test:**

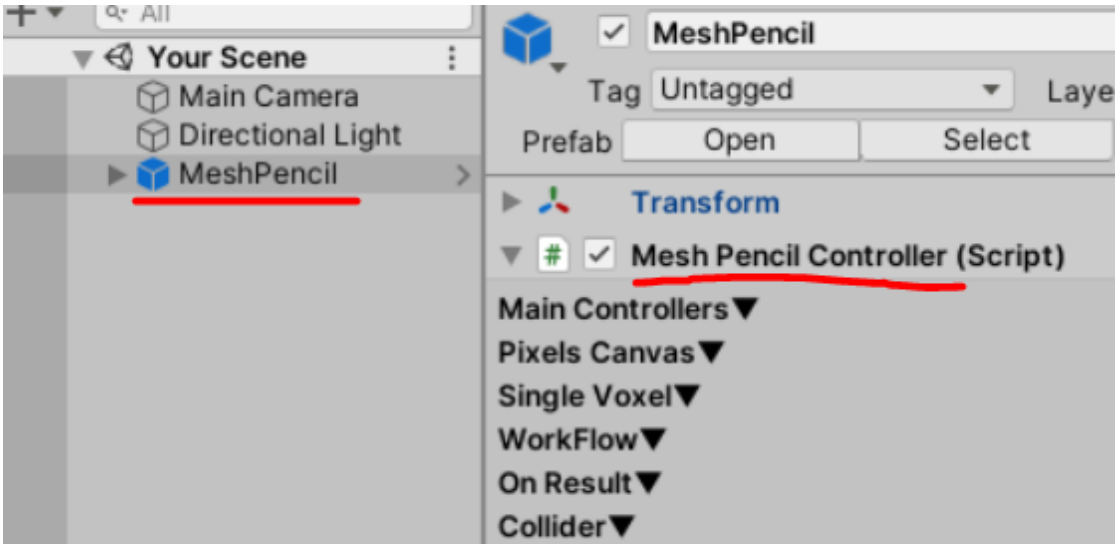
4)Draw something in play mode and press [Space] button to create a mesh




Success result

# Usage interface

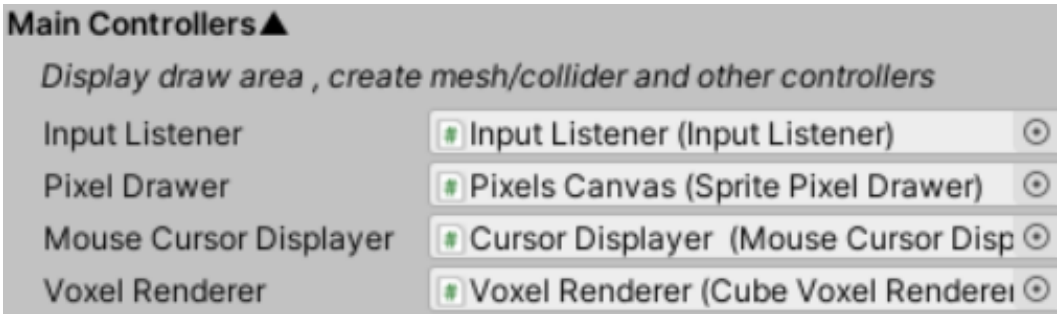
The main controller component for Mesh Pencil asset contains on the prefab gameobject named as "*MeshPencil*"
which you could found in:
"*Assets/MeshPencil/Prefabs/*" folder




## TABS

### Main Controllers

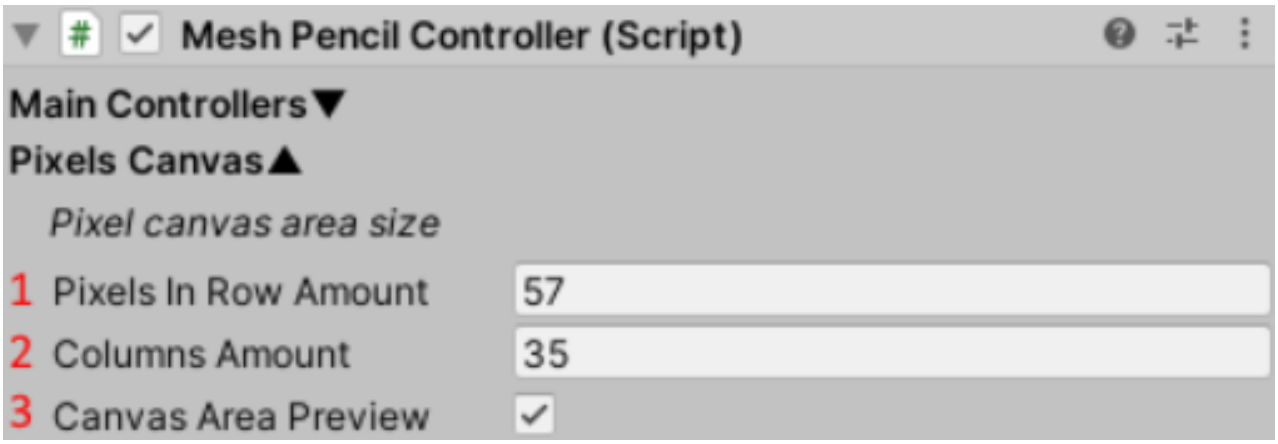
contains links to other controllers :
input system, pixels canvas display, mouse cursor behaviour, final mesh renderer ...
Do not recommend changing this tab.

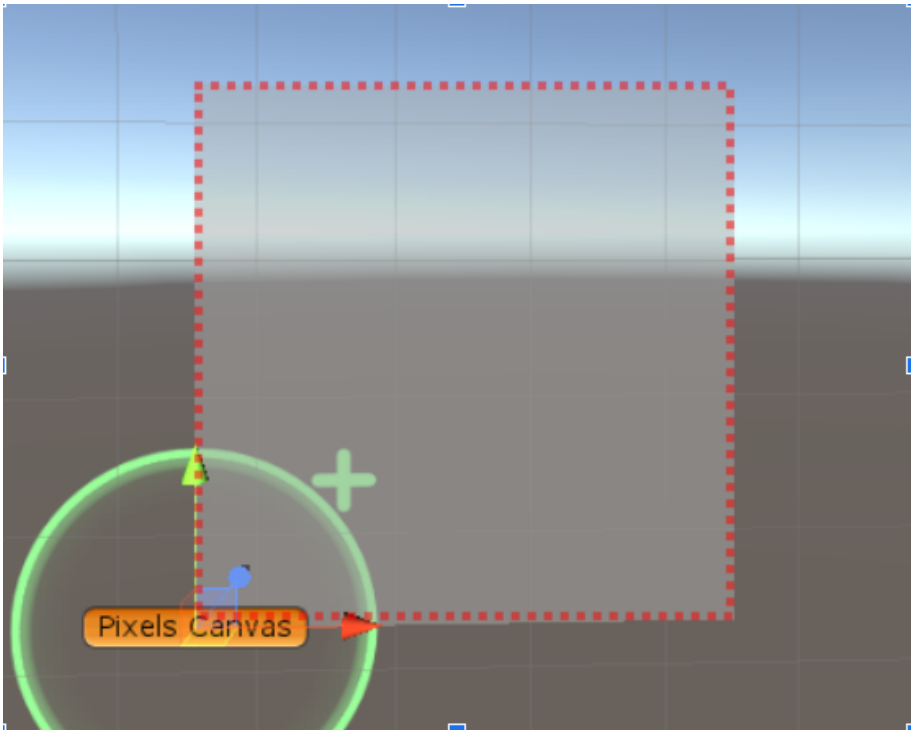


**Pixel Canvas**

controls draw area resolution

1)Number of elements in the horizontal row
2)Number of columns
3)Enable the gizmo in scene view which displays canvas dimensions



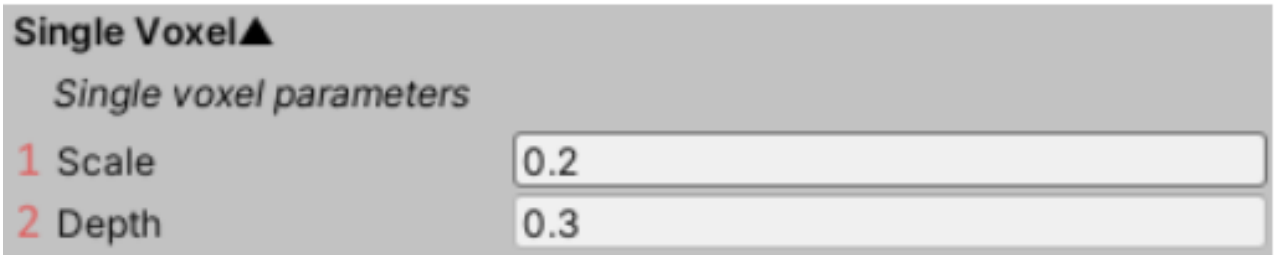




Canvas Area Preview enabled

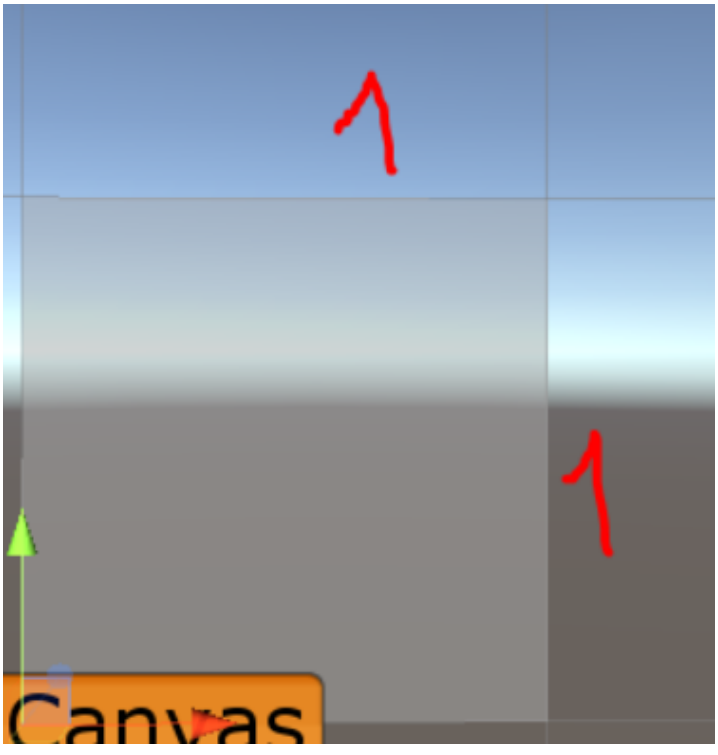
## Single Voxel

controls one single voxel cube parameters

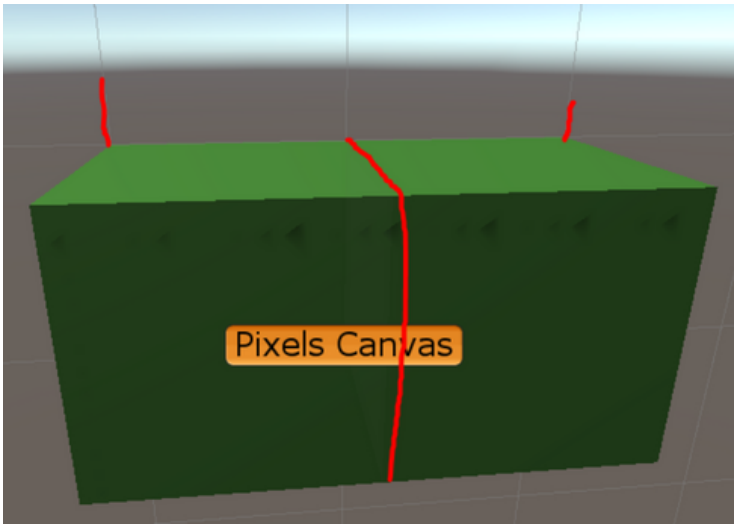
1)Scale by X and Y of single voxel
2)Depth of single voxel (will be multiplied to front and back directions)




Scale 1 equal to 1 cell




Depth 1 equal to 2 cells

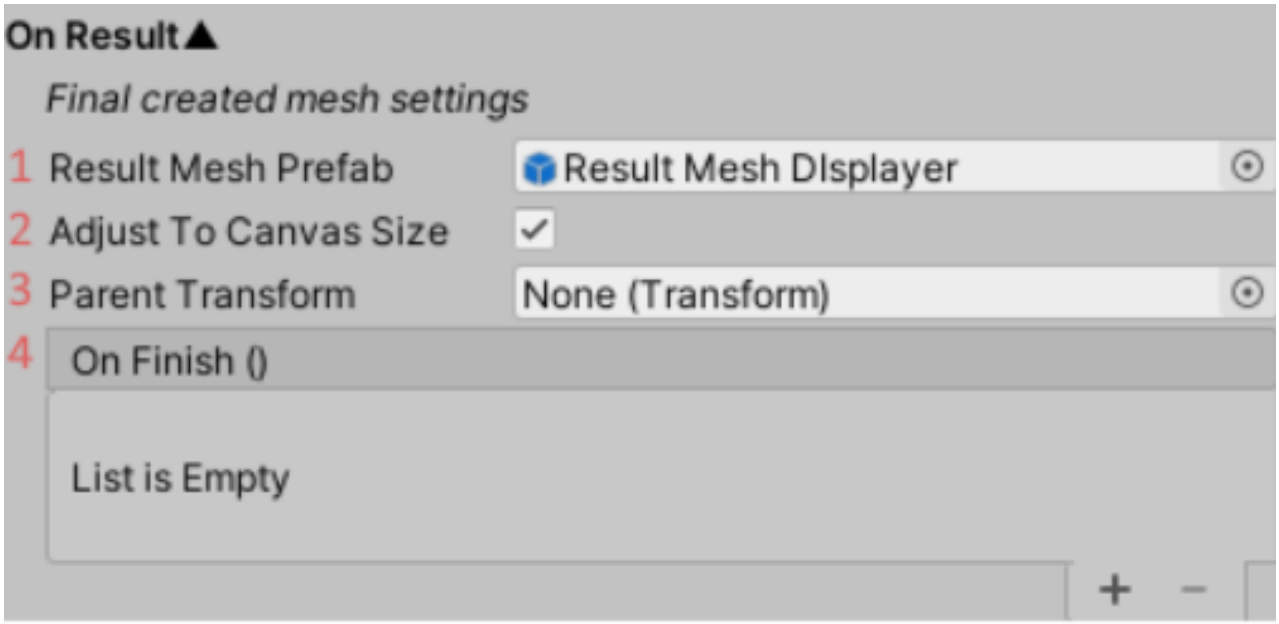


## Workflow

1)IsMultipleDrawMode
True: Enable the endless attempts
False: Disable the canvas after draw finish

## On Result

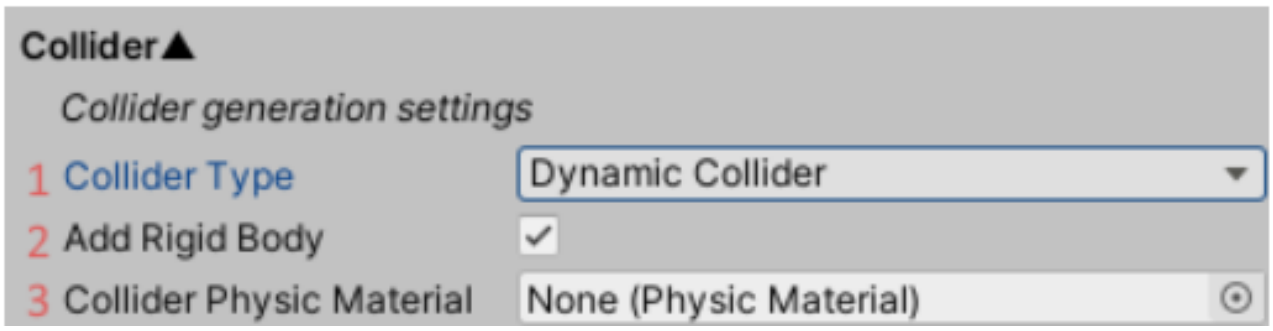
1)Link to prefab which will be instantiated as mesh container
2)Resize and move created mesh to exact canvas position where it was drawn
3)Set a parent of a created mesh object
4)Unity event be called when mesh creation pipeline will finish




You can set your own material of created mesh in the ResultMeshDisplayer prefab, Mesh Renderer component

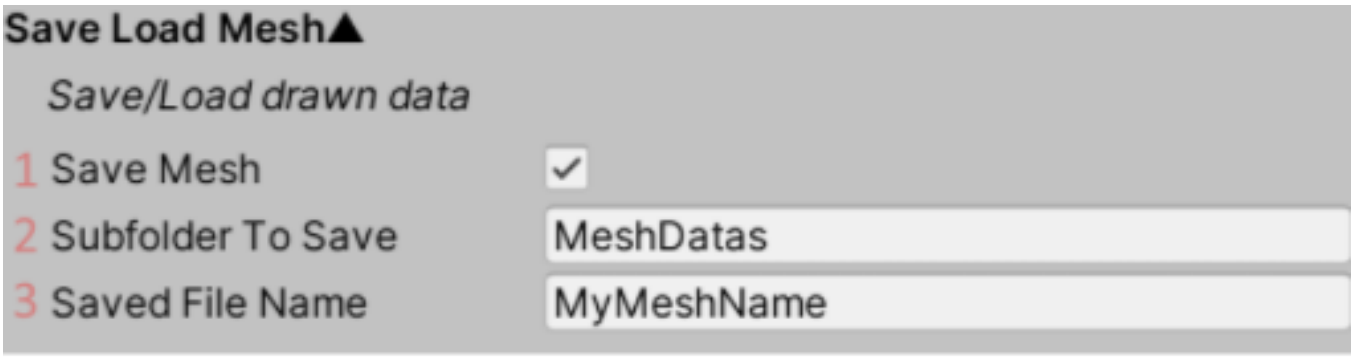
## Collider

1)Type of created collider
None: don`t create any collider
Static Collider: add the mesh collider component (useful for behavior)
Dynamic Collider: generate an array of box colliders for supporting rigidbody component mechanics
2)Add the RigidBody component
3)Add the physics material to collider




## Save Load Mesh

This tab saves your last created mesh to a data file. For more flexible save\load functionality, read "*Base components*" article
1)Enable/Disable automatic mesh save
2)Subfolder name in persistentData
3)Saved data file name

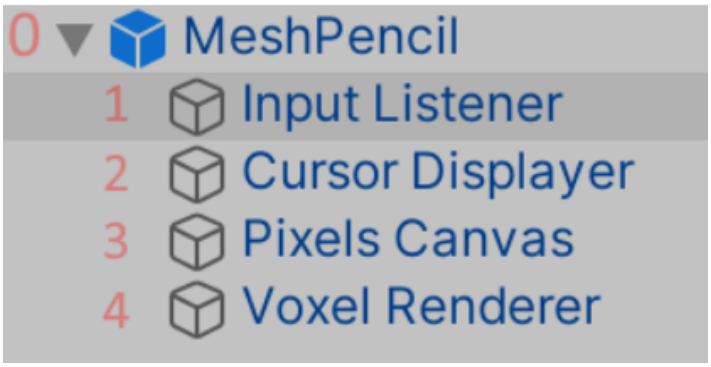


# Base components

**MeshPencilController** is a facade class with common public methods and events :

- UnityEvent OnFinish - invoke when all creating mesh pipeline is finished
- void SaveMeshData(string savedMeshFileName) - save last created mesh data to file. The file will be saved into persistentDataPath.
- void LoadMeshData(string savedMeshFileName) - found and load mesh data by name and creates new mesh.
- void LoadLastCreatedMeshData - found and load last created mesh data by name which controls from MeshPencilController component Save/Load tab

There are other controllers that support common functions “1-4”



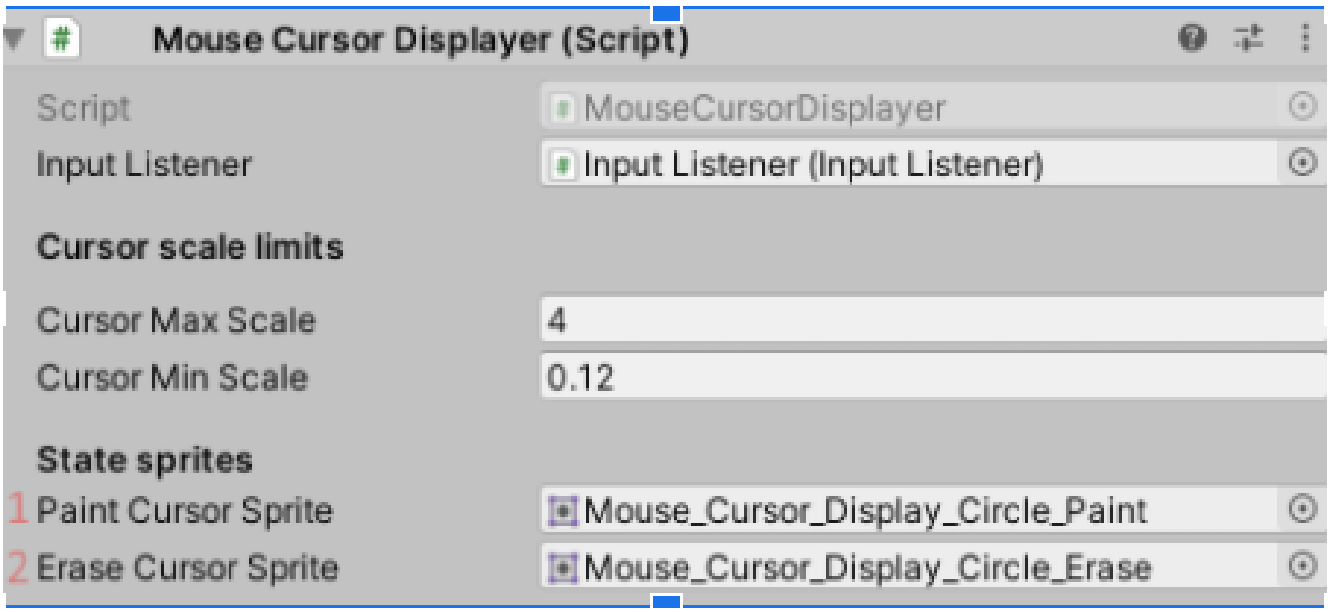
**Input Listener** controls input signals for next commands :

- Paint (start, finish)
- Erase (start, finish)
- Cursor area radius change
- Finish drawing
- Remove all created meshes on scene

**Public methods**

- SetErasingMode(bool isEnabled) : enable or disable erase mode.
  - When erase mode is enable, draw input change painted pixels to unpainted
- GetMousePosition()
  - Returns Vector3 current mouse position
- FinishDrawing()
  - Finish drawing process (useful for mobile input)

**MouseCursorDisplayer** controls cursor radius and sprite:
1 and 2 are fields for paint/erase state sprites.

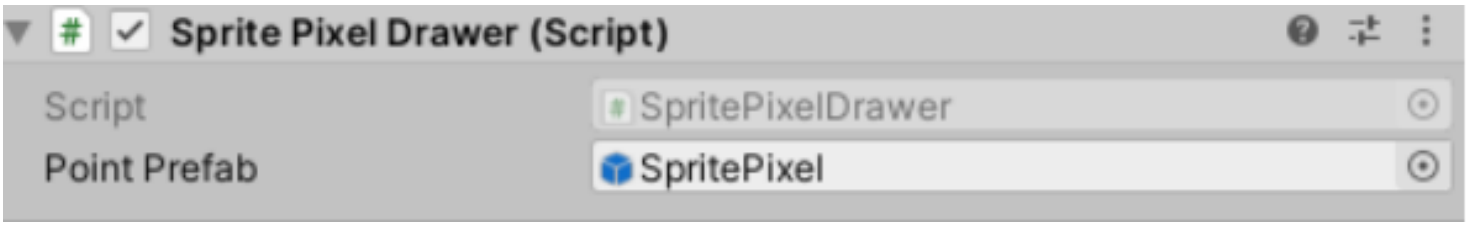



You could create your own cursor sprite.Cursor sprite requires 200x200 resolution.
Don't recommend to use other sprite sizes.

## Public methods

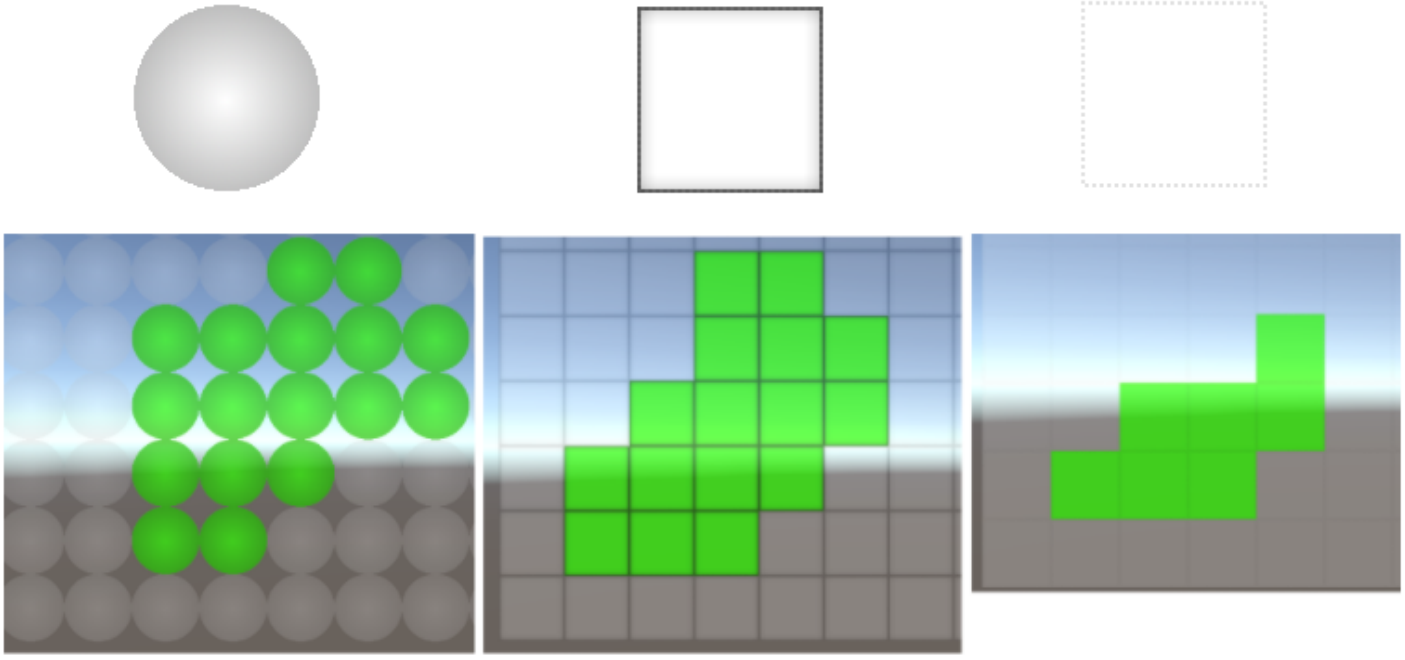
- void ScaleCursor(float scaleAmount)
  - Scale up or down cursor radius (useful for mobile input)

**SpritePixelDrawer** creates area from pixels with configures sizes
(controls from **MeshPencilController** component)
has single fields for pixel prefab



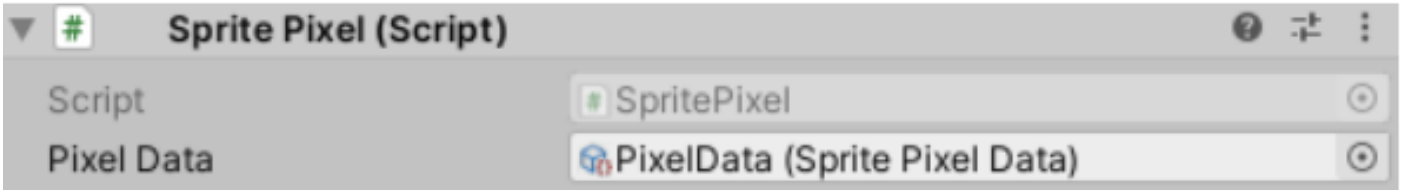

**Pixel customization :**
Open the SpritePixel prefab inside this folder :
*Assets/MeshPencil/Prefabs/Drawer/*

The SpritePixel object has 2 base components for modification :
1)SpriteRendererCould be changed sprite for another sprite with sizes 100x100px

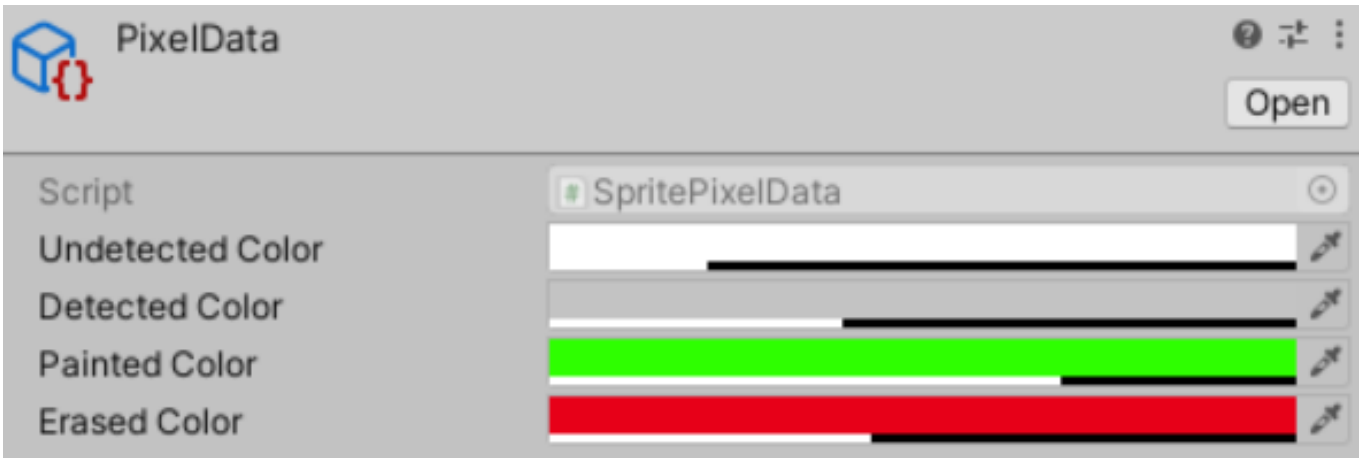


different pixel sprite example

2)SpritePixel->PixelData




also could be founded inside next folder :
*Assets/MeshPencil/ScriptableObjects/*
PixelData is a “scriptable object” data file that contains color presets for pixels states.



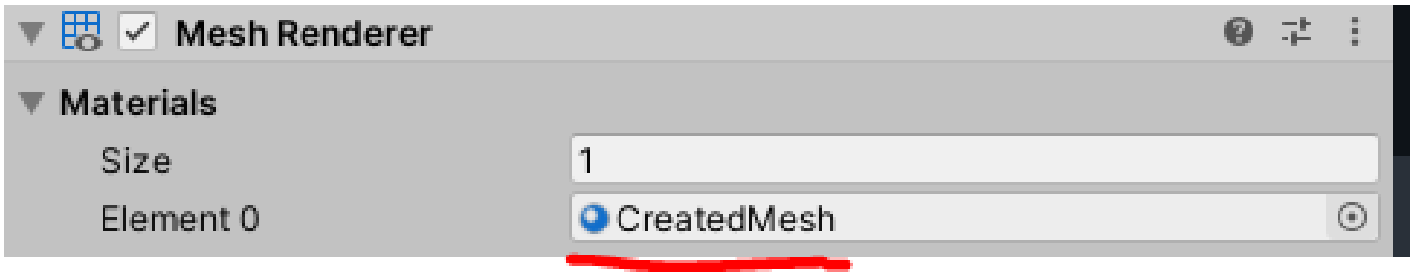

**Public events**

- DrawFinished returns byte[,] when drawing is finished
- DrawFinishFailed returns string error message if draw finish had errors

**4)VoxelRenderer** creates mesh from canvas pixels data.
Result mesh prefab could be changed from MeshPencilController component OnResult tab.

The Result Mesh DIsplayer prefab contains inside this folder :
*Assets/MeshPencil/Prefabs/Renderer*

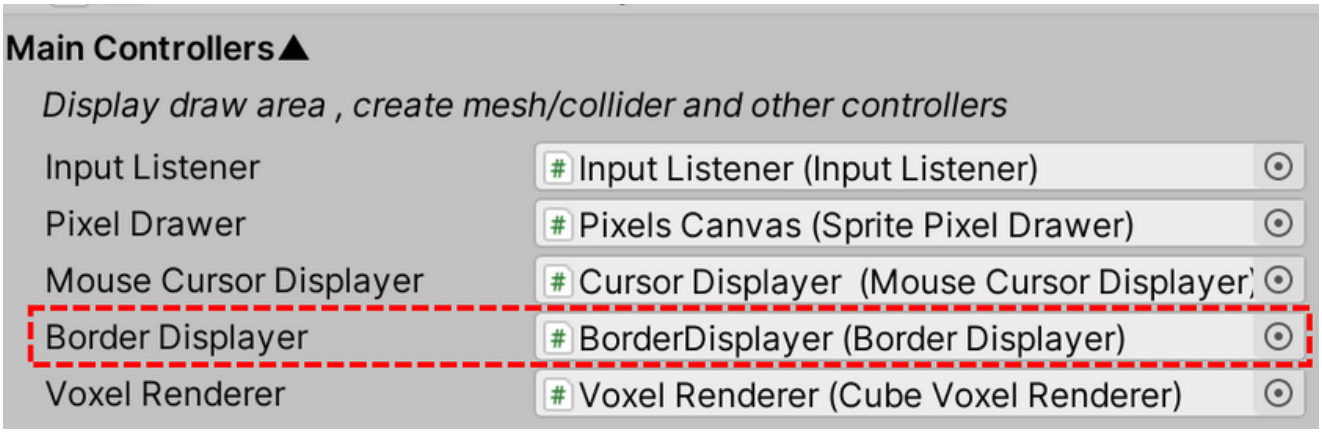
**Result Mesh Displayer** has a MeshRenderer component where you could change material to your own.



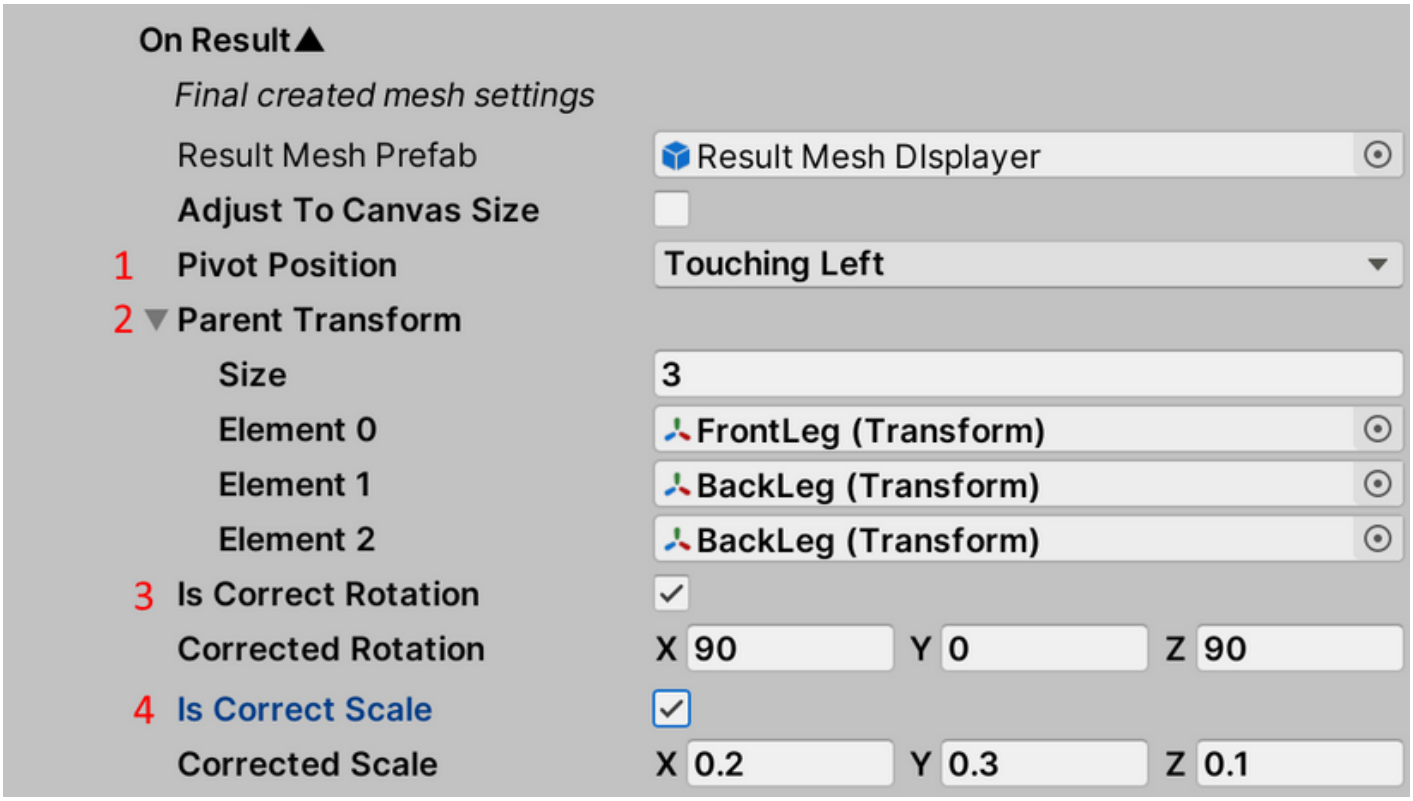

Is able to add other components to final mesh prefab if you need.
Don't recommend removing **CubeVoxelColliderCreator** component, it could break collider create functional.

# 1.2 Update features

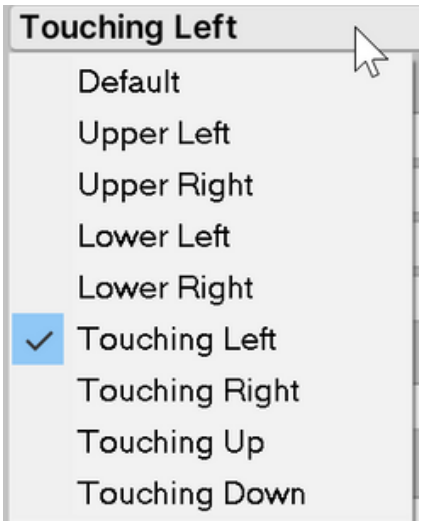
1)**New component for pivot works**.
BorderDisplayer creates pivot point and set a created mesh as a child to this object.




Controls for this component was added into MeshPencilController's OnResult tab

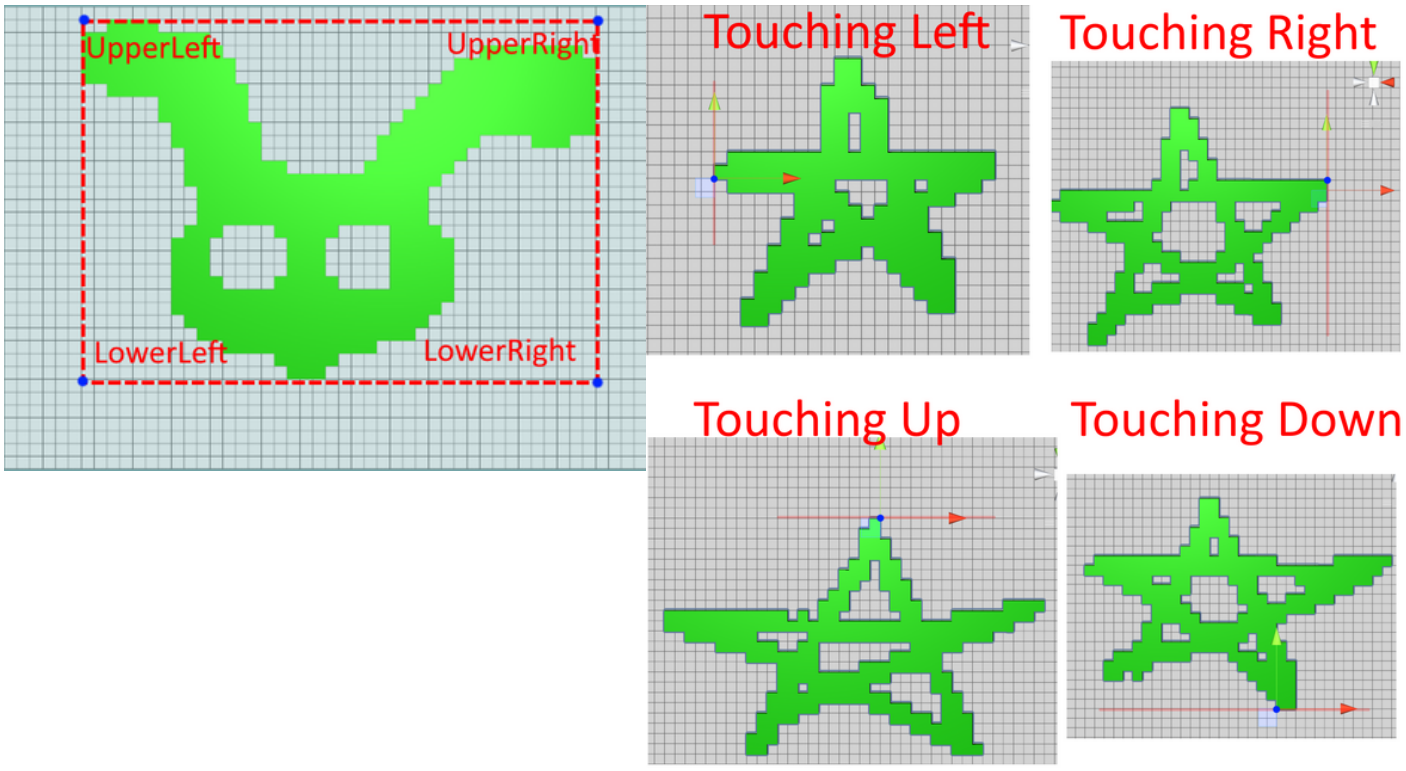


1)Set created mesh as a child to created pivot object. Where to create pivot object you could choose in dropdown




Default option is not create any pivot object , mesh will be spawned without correction.

Next 4 options create pivot on border points



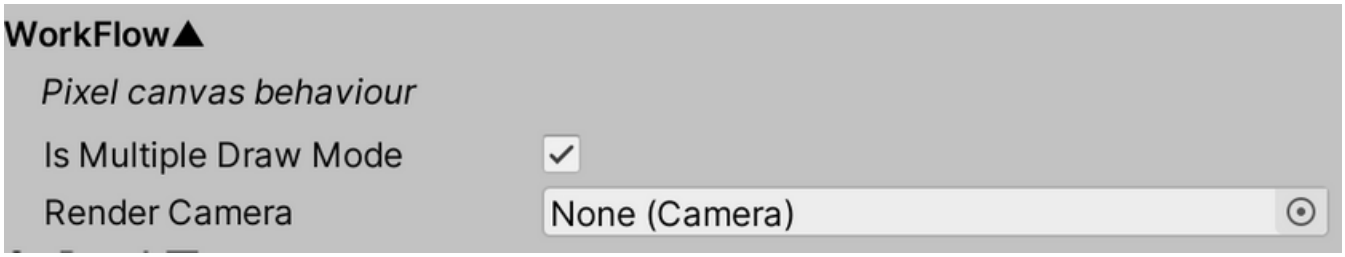

Touching points create pivot on the exist mesh border

2)Pivot parents array.
Now is able to add more then 1 parent.
Mesh will be created once and duplicated to other transforms

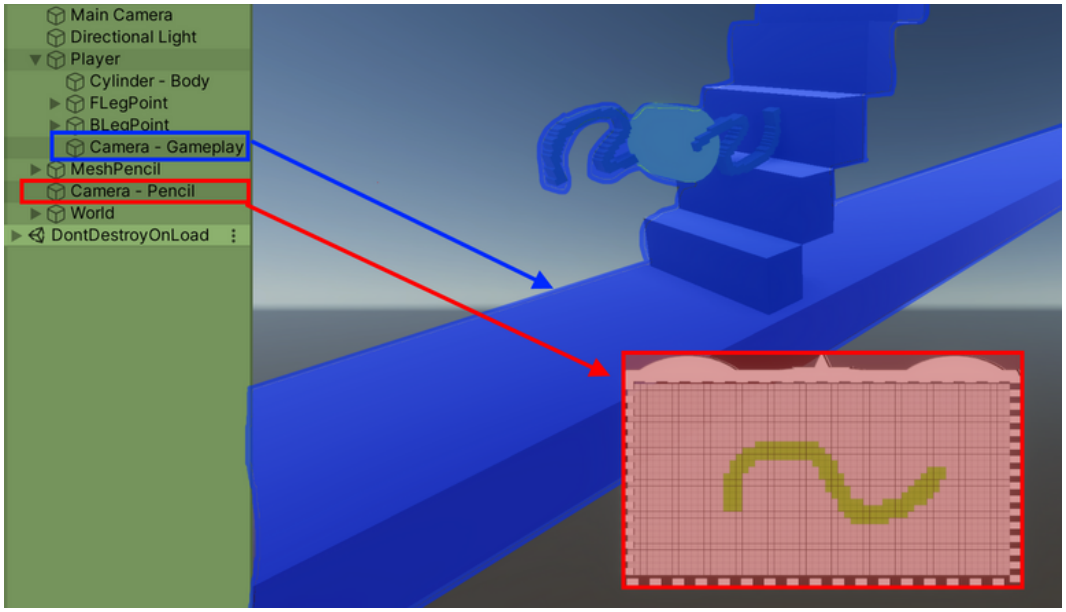
3-4)Correct rotation and scale of a final object

## 2) Render camera

The asset supports render from other cameras for creating more flexible mechanics.
Target camera field is able in MeshPencilController Workflow tab




Leave this field null if your Mesh pencil renders from single camera(*Camera.main*)



## 3)Events OnFinish & BeforeMeshSpawned

Action<GameObject> OnFinalObjectSpawned;
Returns spawned mesh object

Action BeforeMeshSpawned
Invokes before mesh is spawned , but after it is drawn

## 4)Public Settes and Getters

**Setters :**

SetCanvasResolution(Vector2 resolution,float pixelScale)

SetDepth(float depth)

SetMultipleDrawMode(bool isEnabled)

SetResultMeshPrefab(GameObject resultMeshPrefab)

SetResultParentTransform(Transform[] parentTransforms)

SetResultRotation(Vector3 rotation)

DisableRotationCorrection()

SetResultScale(Vector3 scale)

DisableScaleCorrection()

SetPivotPosition(PivotPosition pivotPosition)

IsAddRigidbody(bool isAddRigidbody)

SetPhysicsMaterial(PhysicMaterial physicMaterial)

SetColliderType(ColliderType colliderType)

**Getters :**

GetCanvasResolution

GetPixelScale

GetPixelDepth

IsMultidrawMode

GetRenderCamera

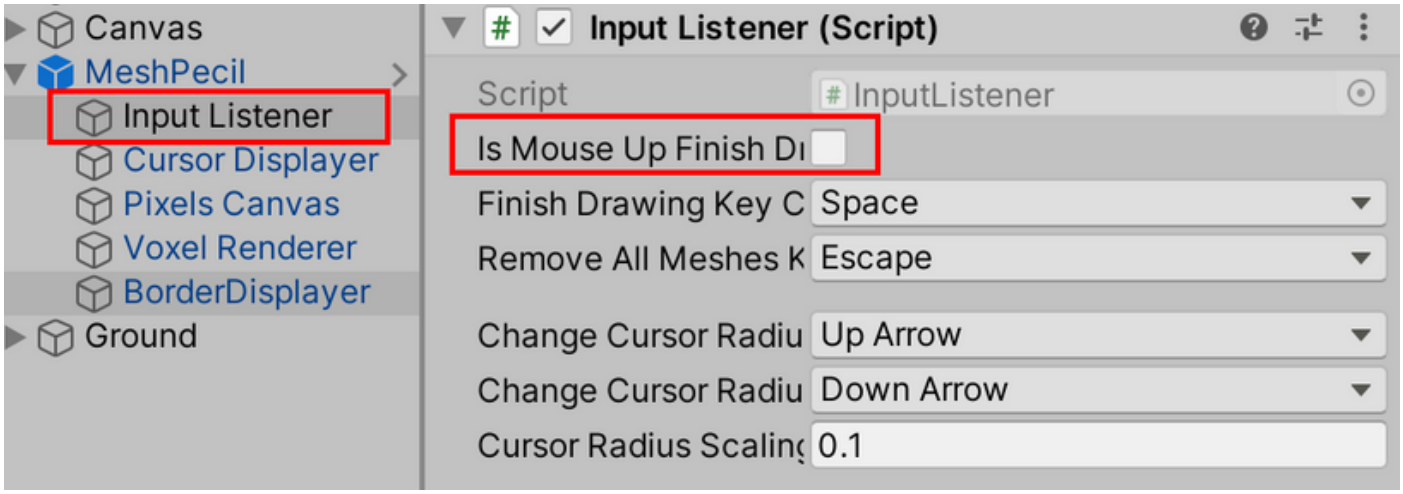
GetPivotPosition

GetColliderType

# 1.3 Update features

1)Draw to OnMouseUp event

Enable the checkbox on the "Input Listener"

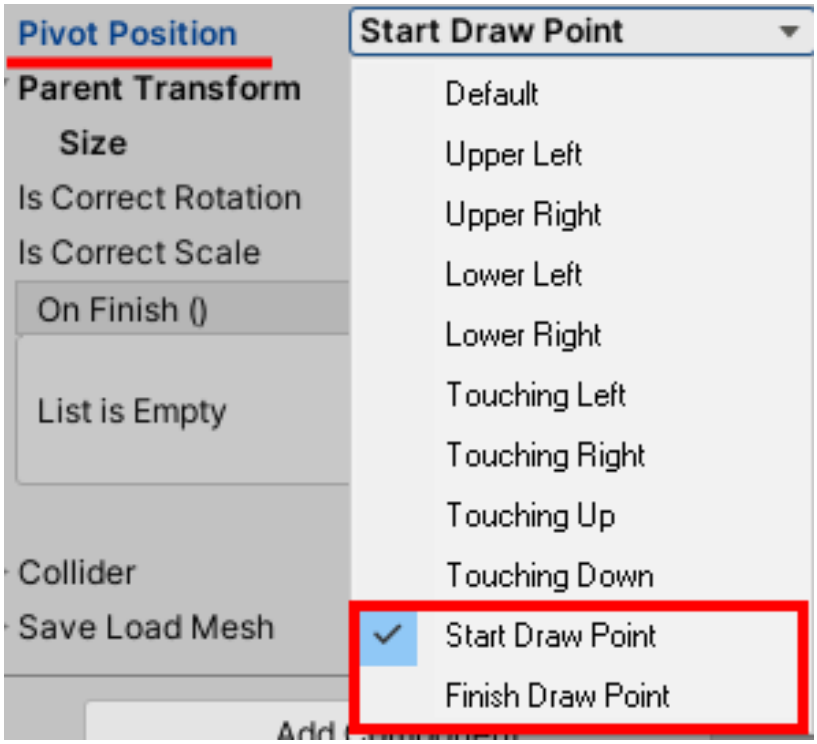



**Finish drawing key will be ignored if checkbox is enabled**

2)Start/finish drawing pivot

Applying the pivot to start/finish drawing point

**Main controller OnResult tab**



# Demo scenes

All demo scenes are placed in next folder:
*Assets/MeshPencil/Scenes*

Demo scenes created for demonstration of sceneries where and how the asset could be used.
We **don’t recommend** use code from demo scenes in the production build.
All code solutions are quick and simple.

**Demo** is a basic scene with a demonstration of core functionality.

**Controls :**

- Left mouse button down - paint
- Right mouse button down - enabling erase mode
- Mouse wheel scroll - change cursor radius
- Space - finish painting

Also, the scene has UI panel with all basic input controls (useful for mobile input)

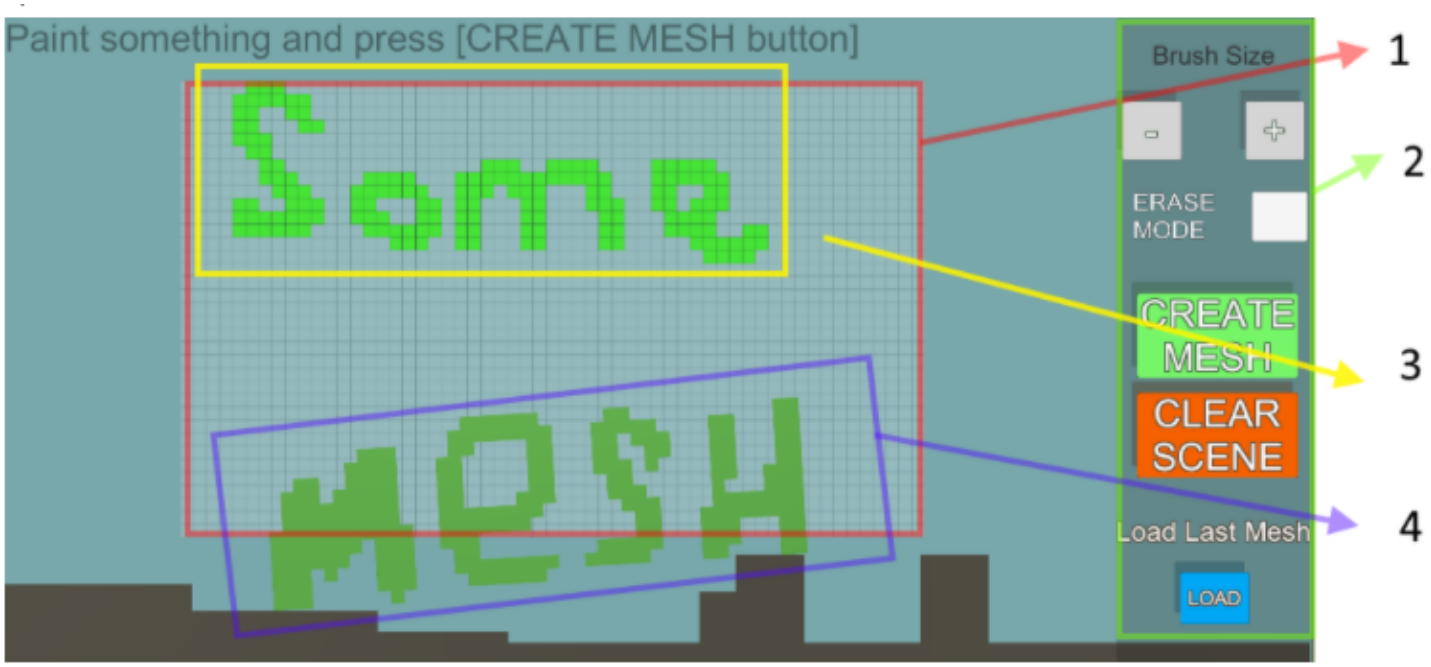
Paint something on pixel canvas and press [SPACE] button to create a mesh.

1)Canvas area

2)UI input panel

3)Painted pixels

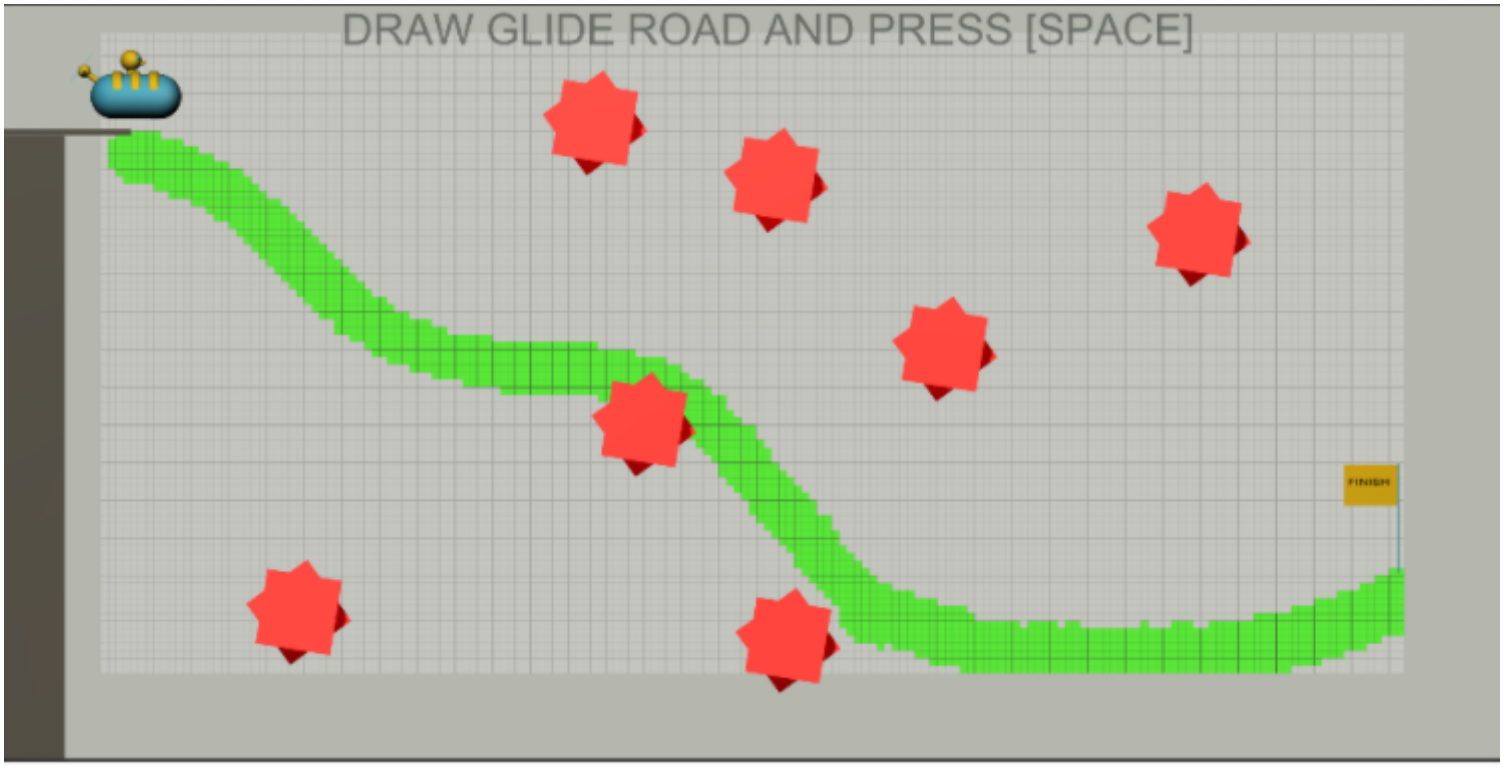
4)Created mesh

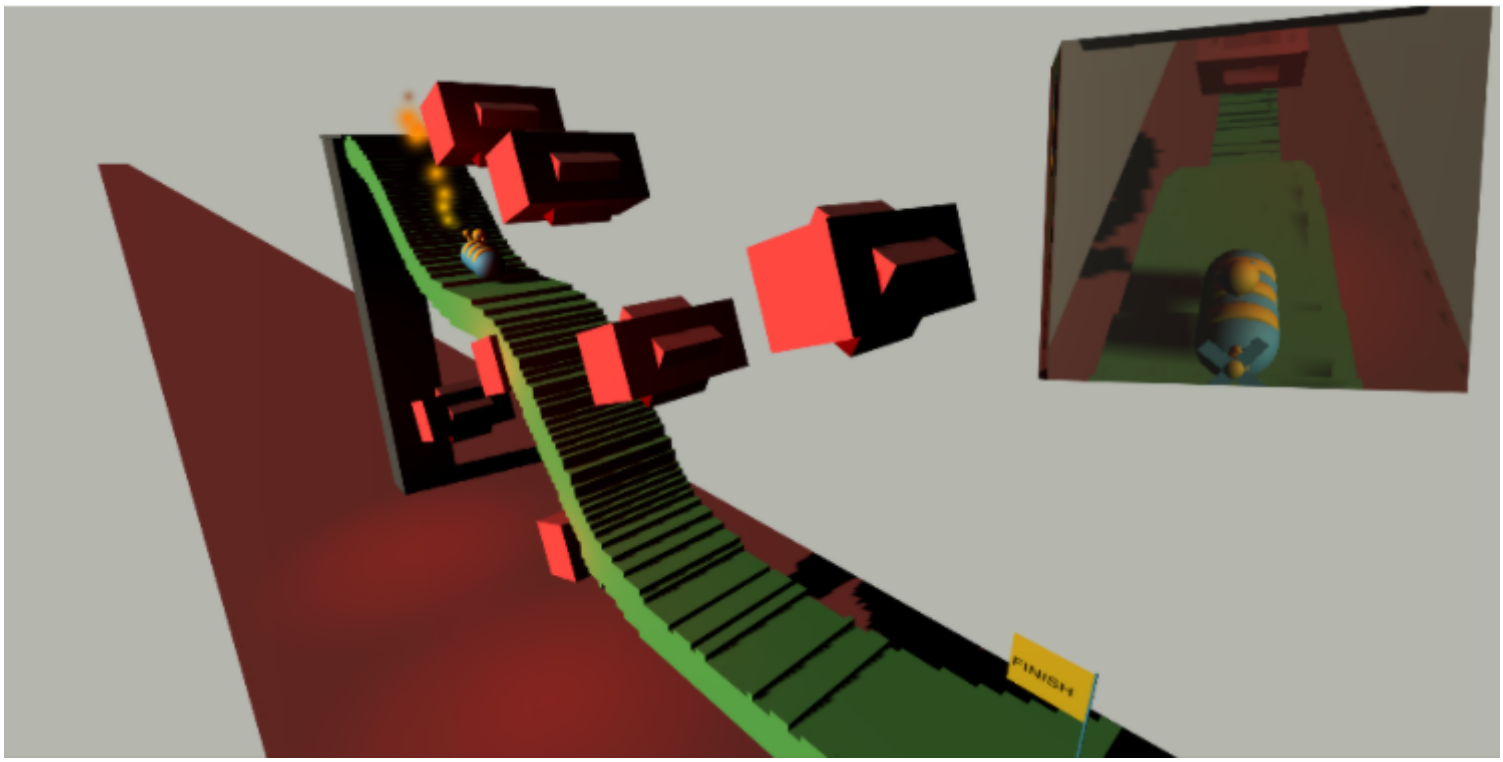


**Rolled Ball**

A scene with classic game where you need transfer the player to finish point

Draw the line for player gliding and press [SPACE] button

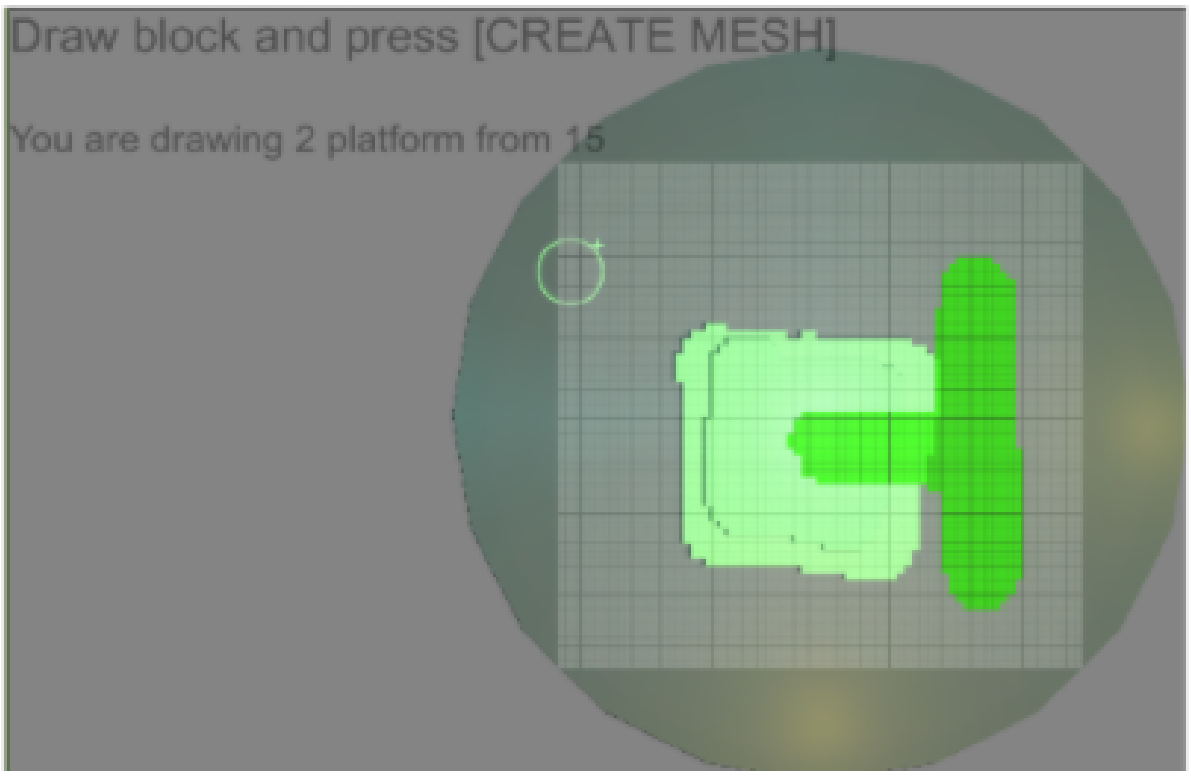


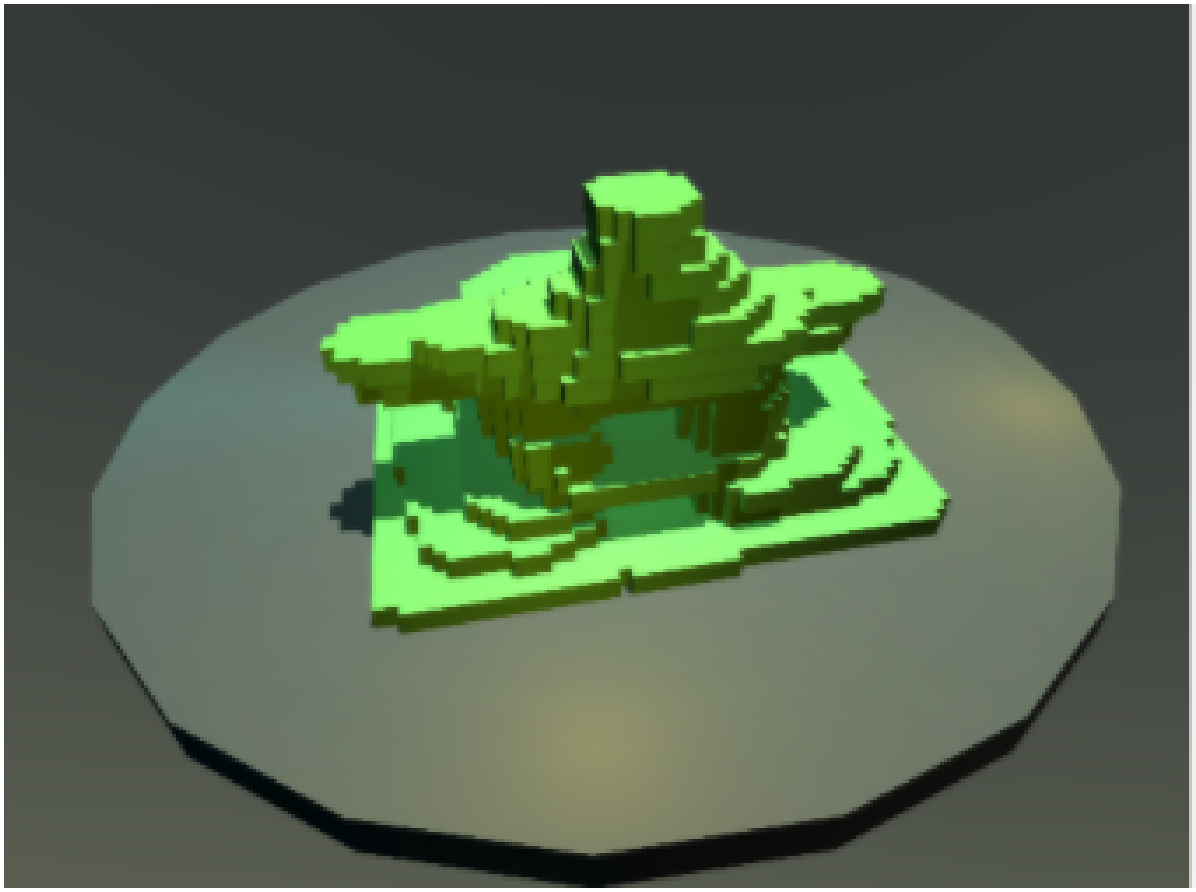



**DrawTower**

Draw an object layer by layer.

Draw current layer and press [SPACE] for a move to next layer

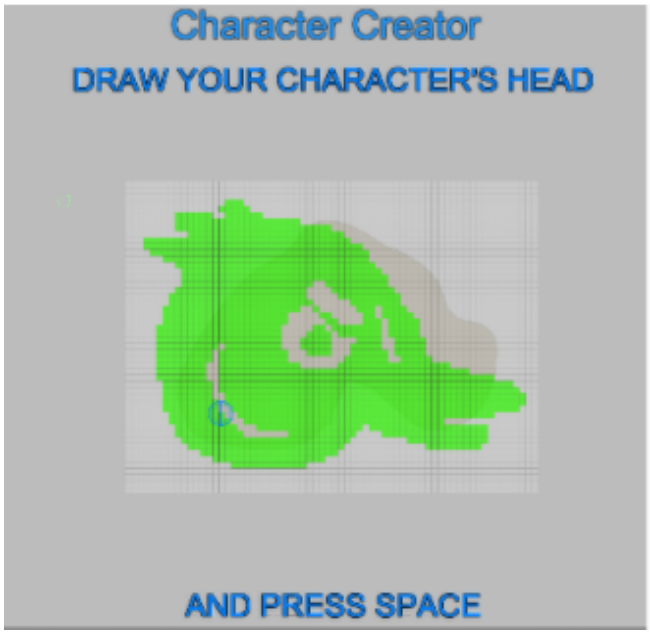


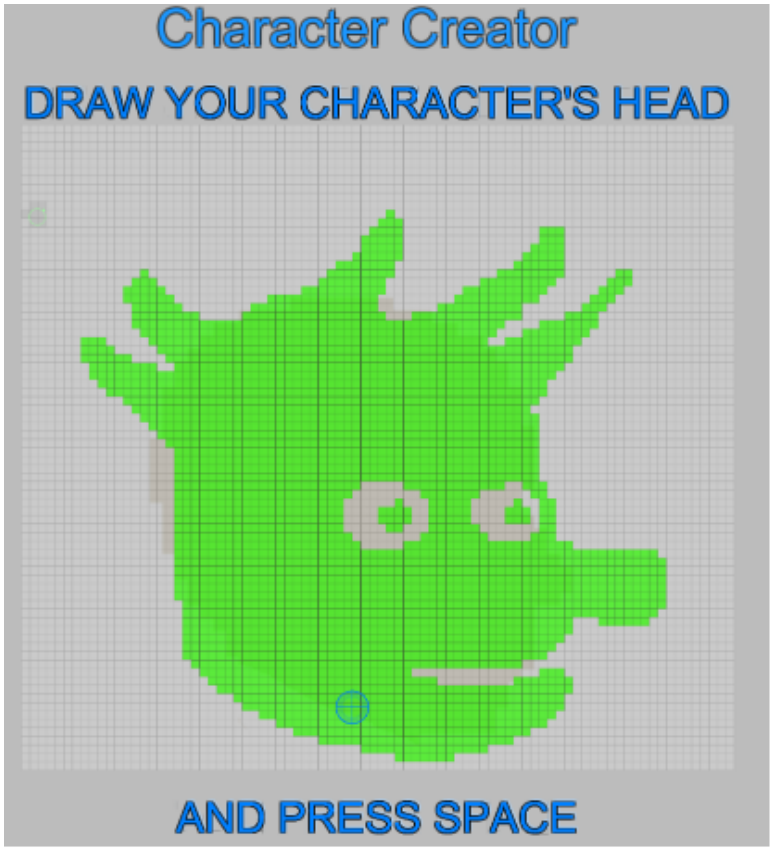



**CreateAnimalDemo & CreateCharacterDemo**

Constructor scene where you draw all single parts of body and get a creature, based on your drawn data.

Draw the current body part and press [SPACE] button to move forward




# Additional tutorials

ASSET DEMONSTRATION VIDEO

https://www.youtube.com/watch?v=-Z7_YB4POfk

USER MANUAL VIDEO TUTORIAL

https://www.youtube.com/watch?v=49e8F4Fi8aA

HOW TO SETUP RENDER CAMERA

https://www.youtube.com/watch?v=12PAgIq9WRs

SIMPLE DRAW MECHANIC GAME

https://www.youtube.com/watch?v=8gjyEBPVs0s

HOW TO USE PIVOTS, PARTENTS

https://youtu.be/kzLCpHBVWn8

# Contacts

Skype : germetic1
@mail : germetic14@gmail.com
discord : Nako Rinn#5314