

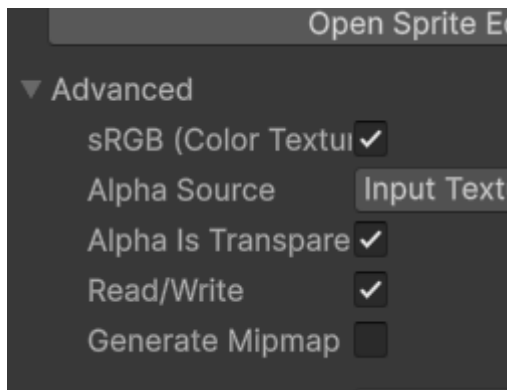
Destro 2D:

Overview:

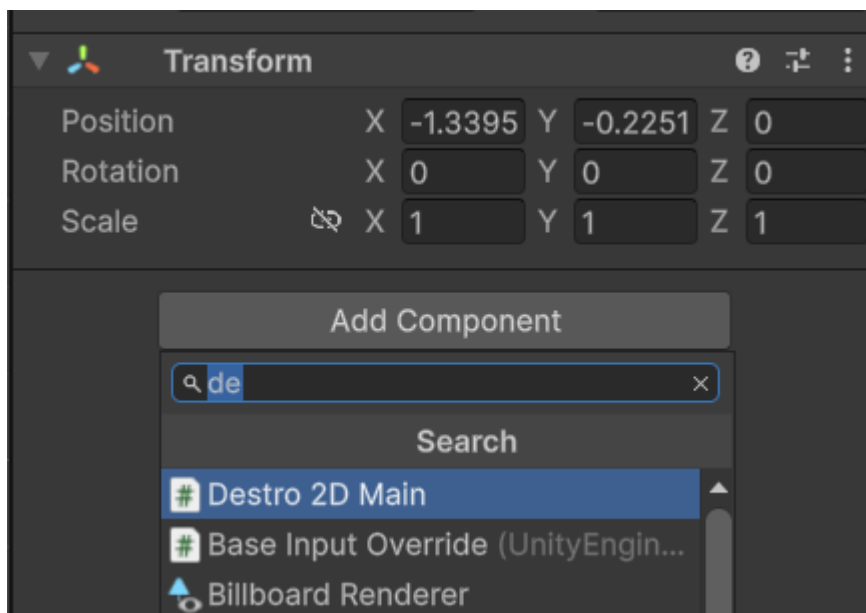
1. Dynamically destroy sprites(stamp them with textures, or destroy with predefined shapes)
2. Collision updates automatically
3. Split regions have their own Game Objects + physics

Quick Setup:

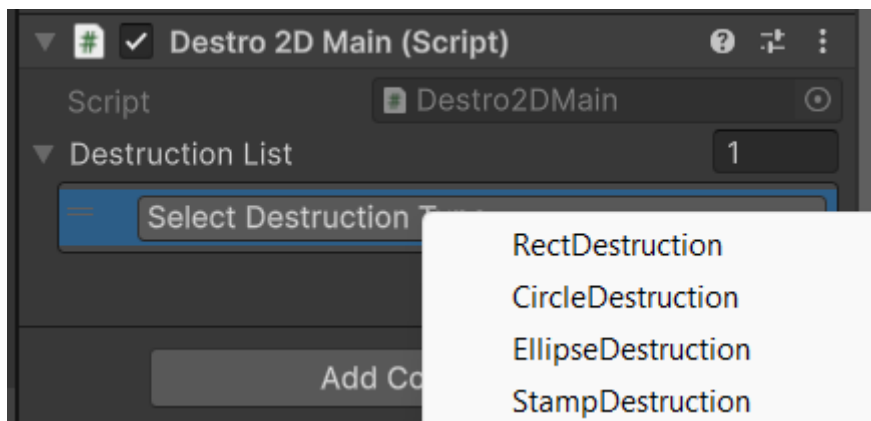
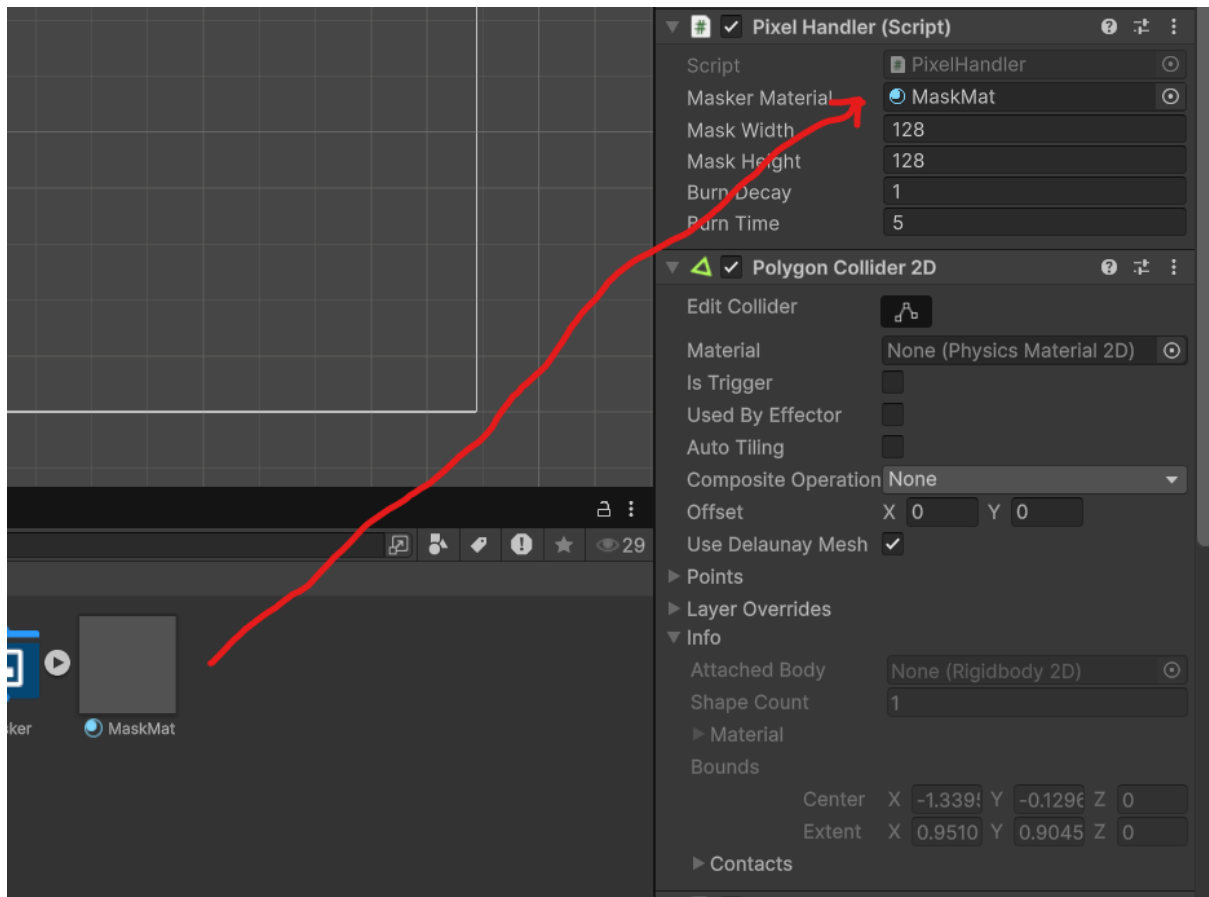
IMPORTANT: Make sure your sprite has Read/Write enabled (ON) in the import settings, then click Apply



Add Destro2DMain component to your Sprite

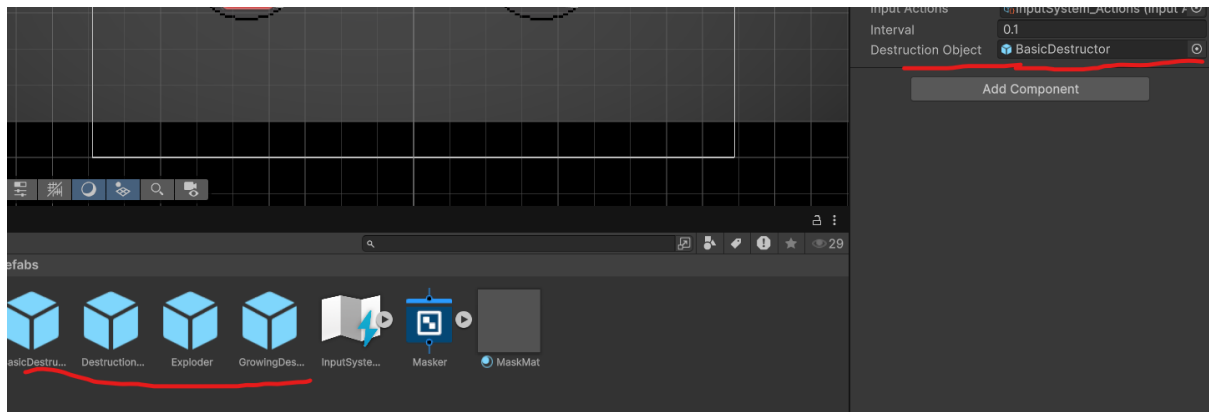


Set up the material (mask material), then select the destruction type your Sprite will be susceptible to. This can be changed dynamically, which will be explained later.



Drag and drop the DestructionSpawner Prefab onto scene, then select the appropriate Destructor Prefab to put in the Destruction Object field. The interval decides how often the object gets spawned onto the scene.

Alternatively, you can directly instantiate the Destructor Object on top of a Sprite to apply destruction at that location. DestructionSpawner simply spawns the Destructor Object on mouse click.



Run Game, Click on Sprite.

Detailed Use:

Main Components:

PixelHandler – Handles visual updates

CollisionHandler – Updates collision based on visual changes

SplitHandler – Detects and generates chunks based on visual changes

Destro2DMain – Initialises the three above components and provides high level functions for destruction

Sub Components:

Masker – Shader Graph that implements the masking, along with the burn

MaskMat – Material Instance of Masker; create your own as you please, edit values such as Burn params etc

Terms:

Mask – it is the 2D Boolean array that encompasses the sprite, all the transformations are done on these. There are three Masks – Visual, Collision and Split. Destruction is done on VisualMask.

Fields:

Pixel Handler -> Mask Width, Height determines width and height of 2D array respectively,

Burn Decay determines how fast the burn decays

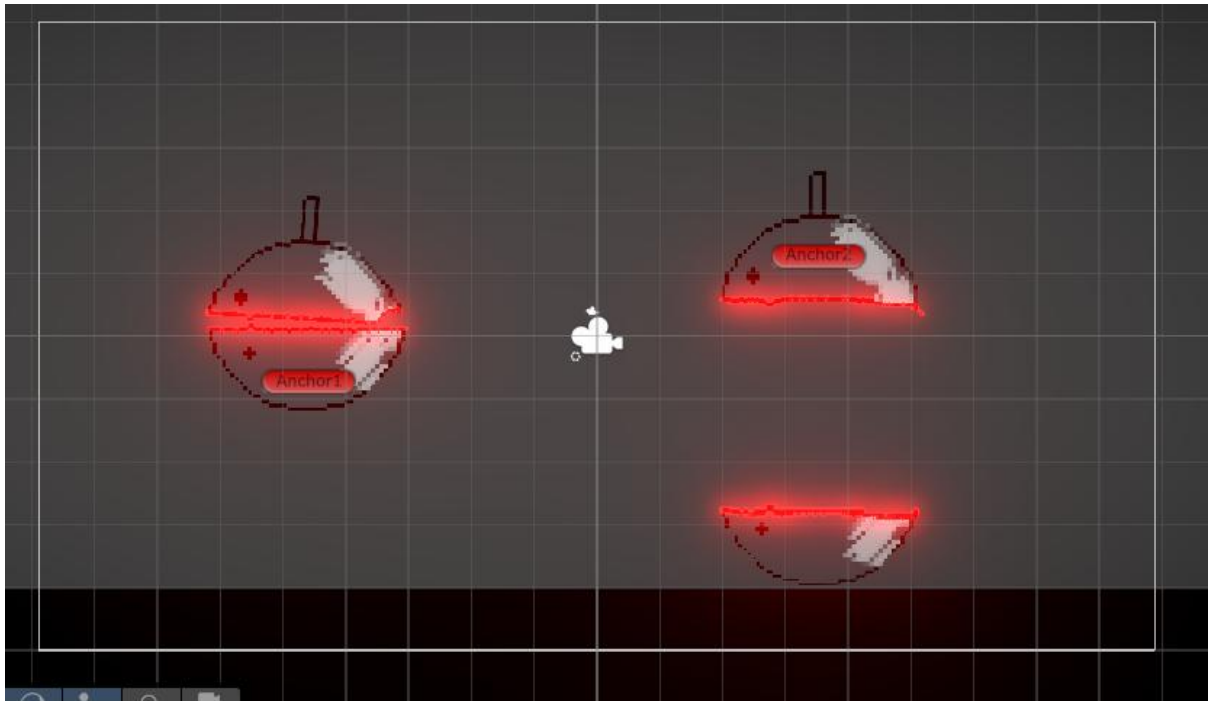
Burn Time decides the time burn lasts.

Note: Burn is only updated on destruction

Split Handler -> MinCount determines the minimum size of block to be split into its own chunk

Chunks can be accessed via the list Destruction.chunks

Anchor determines where the original object is anchored to.



If Anchor is broken, an Action is invoked once. If it is not set, one will be created automatically.

Anchor does not matter for Dynamic GameObjects(with Rigidbody)

Destro2DMain -> destructionList contains the list of destruction types the sprite will undergo upon destruction

Functions in Destro2DMain:

DynamicDestroyWorld(Vector2 loc) -> Applies destruction at world position

DynamicDestroyLocal(Vector2 loc) -> Applies destruction at local position

DynamicFracture(Vector2 world, float Rcore, float Router, float noiseScale, float thickness, int lines) -> Applies fracture starting from Rcore to Router. Recommend lines at 5. Heavy.

AddStampDestruction(Texture2D stampTex, float scale, float rot), AddCircleDestruction(float radius), AddEllipseDestruction(float radiusX, float radiusY), AddRectDestruction(float l, float b) -> Add various destruction types to destructionList. Alternatively, you can create objects of the Destruction types and add them to the list.

Actions:

OnDestruction -> Invokes once on any call to destroy

OnAnchorBroken -> Invokes once anchor is broken.

Code Stuff:

Adding new Destruction type:

1. Make sure the class inherits from the abstract class Destruction and implements its 2 functions.
2. The destruction is applied on visual mask, passed via Pixel Handler

Performance Notes:

1. Mask width and height determines the performance, mostly. Recommend 128 x 128 for Pixel Handler, and 64 x 64 for collision and split.
2. Adjust the frequency of mask updates via changing the waiting time in CheckDirty() IEnumeration in PixelHandler to get better framerate.