Documentation

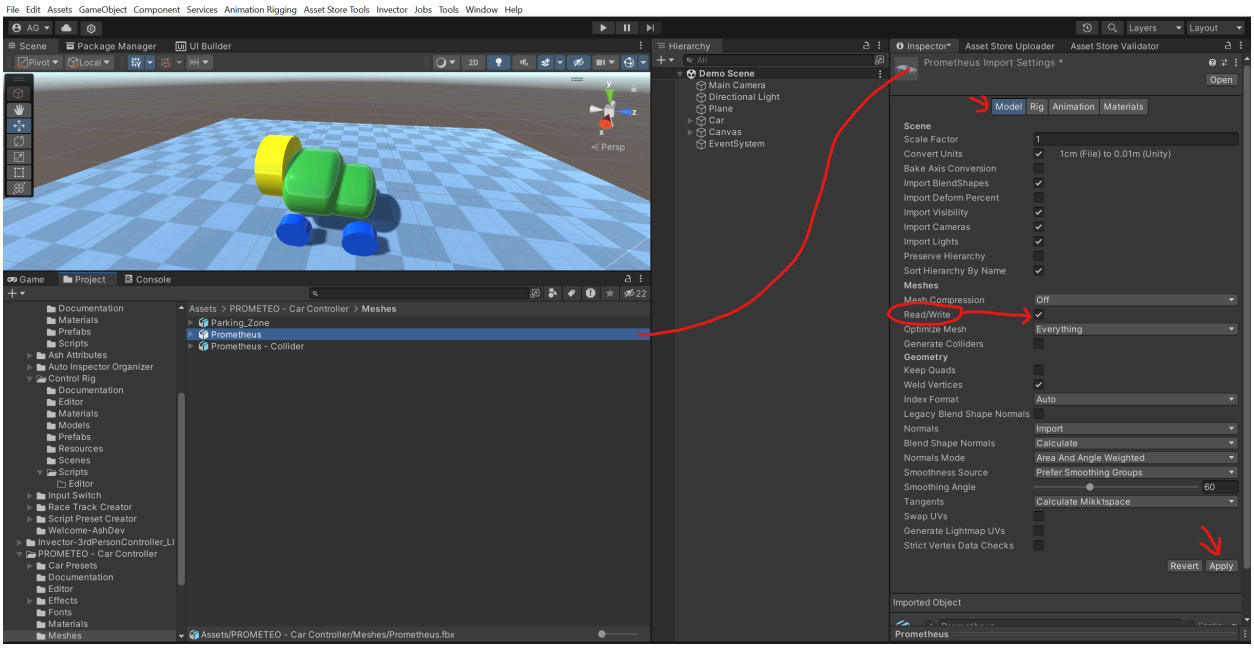
Tutorial video : https://youtu.be/dHTWOclhdPk?si=FdHNXk7CSjMvnOW5

Overview

The **ArcadeDamageSystem** script applies deformations to a object's mesh when a collision exceeds a specified impulse threshold. The deformations are computed in real-time, altering vertices within a defined radius around the impact point. Optionally, the system can update the corresponding mesh colliders to ensure that collision detection remains accurate after deformation.

Prerequisite

make sure that the meshes you want to deform have read and write option enabled. you can set it from the mesh file (like .fbx), then Apply. in the image below i am setting it for a vehicle .fbx model file.



Installation and Setup

1. **Add the Script:**
    - Attach the `ArcadeDamageSystem` script to your GameObject.
2. **Assign Mesh and Collider Components:**
    - **Mesh Filters:** Assign all relevant `MeshFilter` components representing Object parts that will deform.
    - **Mesh Colliders:** Assign corresponding `MeshCollider` components if you want the collision shapes to update with the deformations.
    - Note : first element of `MeshFilter`'s array should be of first element of `MeshCollider`'s array, and so on. So there should be both mesh collider and mesh filter preset on the mesh you want to deform.
3. **Auto-Detection (Recommended):**
    - Use the context menu option **"Detect Child Meshes and Colliders"** to automatically populate the `ObjectMeshFilters` and `ObjectMeshColliders` lists from child objects.

- for this to work, this script should be added on top most parent of the Gameobject.

Script Configuration

Mesh and Collider Settings

- **ObjectMeshFilters**

  Description: List of `MeshFilter` components representing the parts of the Object that are subject to deformation.

  Usage: Manually assign or auto-detect using the context menu.

- **ObjectMeshColliders**

  Description: List of `MeshCollider` components that correspond to the Object mesh filters.

  Usage: Used to update the collision boundaries after mesh deformation. Auto-detection is available via context menu.

- **updateColliders**

  Description: When enabled, the script updates the colliders to match the new, deformed mesh geometry.

  Usage: Toggle on if accurate collision detection post-deformation is required.

Deformation Settings

- **deformationIntensity**

  Description: Controls the strength of the deformation effect applied during collisions.

  Usage: Increase for more pronounced damage; decrease for subtle deformations.

- **deformationSmoothness**

  Description: Determines the smoothness of the deformation. Higher values result in a more gradual transition in the affected area.

Usage: Adjust to achieve the desired visual smoothness in the mesh deformation.

- **deformationRadius**

  Description: Sets the radius of influence around the impact point within which vertices will be deformed.

  Usage: Use the range slider (0.05f to 5f) to define the area affected by the impact.

- **maxDeformationsPerVertex**

  Description: Specifies the maximum number of times each vertex can be deformed. This prevents excessive distortion at a single vertex.

  Usage: Modify based on how resilient you want the mesh to be over multiple impacts.

- **collisionImpulseThreshold**

  Description: The minimum collision impulse required to trigger a deformation.

  Usage: Set this value based on the desired sensitivity to collisions.

Unity Events

- **onDeformationApplied**

  Description: Event triggered immediately after a deformation is applied due to a collision.

  Usage: Attach any additional functionality (e.g., sound effects, particle systems) that should execute post-deformation.

- **onDeformationReset**

  Description: Event invoked when all deformations are reset to the original mesh shapes.

  Usage: Use this event to handle any logic required after resetting the Object's appearance.

How It Works

- **Collision Detection:**

  The script listens for collision events using `OnCollisionEnter()`. If a collision's impulse magnitude exceeds `collisionImpulseThreshold`, deformation is triggered.

- **Deformation Process:**

  The deformation effect is applied through a Burst-compiled job (`DeformJob`) that processes the mesh vertices in parallel. Each vertex within the `deformationRadius` of the impact point is moved along the collision normal, with the amount of deformation scaled by distance, `deformationIntensity`, and `deformationSmoothness`.

- **Mesh and Collider Updates:**

  After the vertex modifications, the mesh is updated, and its normals and bounds are recalculated. If enabled, corresponding colliders are also updated to reflect the new shape.

Usage Instructions

Collision-Based Deformation

- **Setup:**

  Ensure your Object GameObject has the necessary `MeshFilter` and (optionally) `MeshCollider` components assigned.

- **Operation:**

  When a collision occurs and its impulse exceeds the defined threshold, the script automatically applies a deformation effect at the contact point. The deformation propagates over the mesh vertices within the specified radius, simulating real damage.

Resetting Deformations

- **Manual Reset:**

  To revert the Object to its original shape, right-click on the component in the Unity Inspector and select **"Reset Deformation"**. This copies the original vertex positions back to the current mesh and clears any accumulated deformation counts.

- **Event Trigger:**

  Upon resetting, the `onDeformationReset` Unity event is invoked, allowing you to execute any additional reset logic.

Detecting Child Meshes and Colliders

- **Auto-Detection:**Use the **"Detect Child Meshes and Colliders"** context menu option to automatically scan the GameObject's children and populate the `ObjectMeshFilters` and `ObjectMeshColliders` lists. This is particularly useful for complex Objects with multiple mesh components.

Troubleshooting

- **No Deformation Occurs:**
  - Verify that the collision impulse is above the `collisionImpulseThreshold`.
  - Check that the correct `MeshFilter` and `MeshCollider` components are assigned.
  - Ensure that the deformation parameters are set to values that produce visible changes.
- **Collider Issues:**
  - If deformed meshes do not register collisions correctly, ensure that `updateColliders` is enabled and that colliders are correctly linked to the mesh filters.
- **Performance Concerns:**
  - For large or complex meshes, consider reducing the number of vertices.