Castle Climber REDUX

Matt Peterson – DES450 FALL 2023 – Technical Documentation

In-Engine Improvisations & Innovations:

- URP shader manipulation
  - Triplanar mapping
  - 2D and 3D cross integration through shaders and lighting
- Animation manipulation
  - Animation of static mesh through UV manipulation
  - Accessing properties of URP shader materials for keyframing
- 2D backend behind 3D frontend
  - Execution of 2D physics/controller/tilemapping behind 3D
  - Working in a 2D workflow masked by a 3D presentation
  - Blending the difference between 2D and 3D visually
- Modular player controller design
  - Utilizes Finite State Machine logic to adapt expandability and modularity
  - Easily built upon; features can easily be added or removed
  - Allows locomotion logic to be split between different branches of functionality, across different scripts and states.

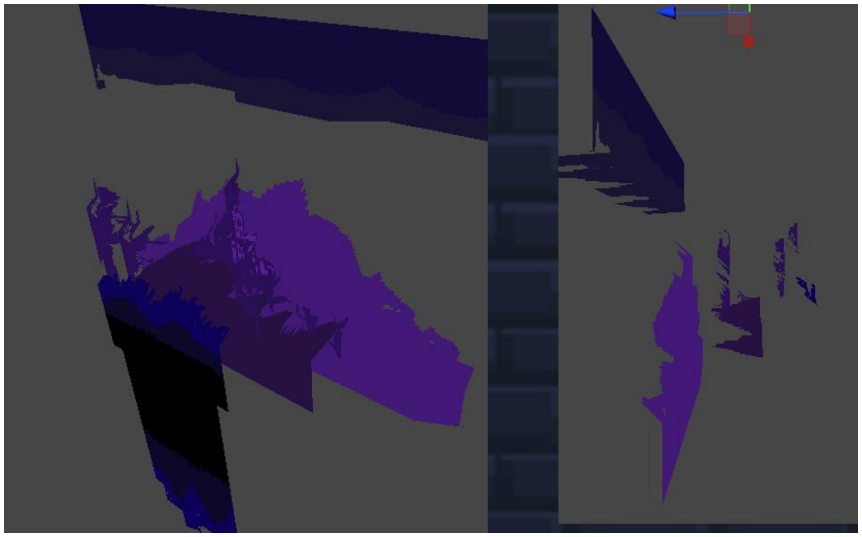
Demonstrative Examples:

Triplanar shader projecting the brick texture across UVs, using custom shader + materials per texture projected. Normal of object determines which texture is projected across which plane.

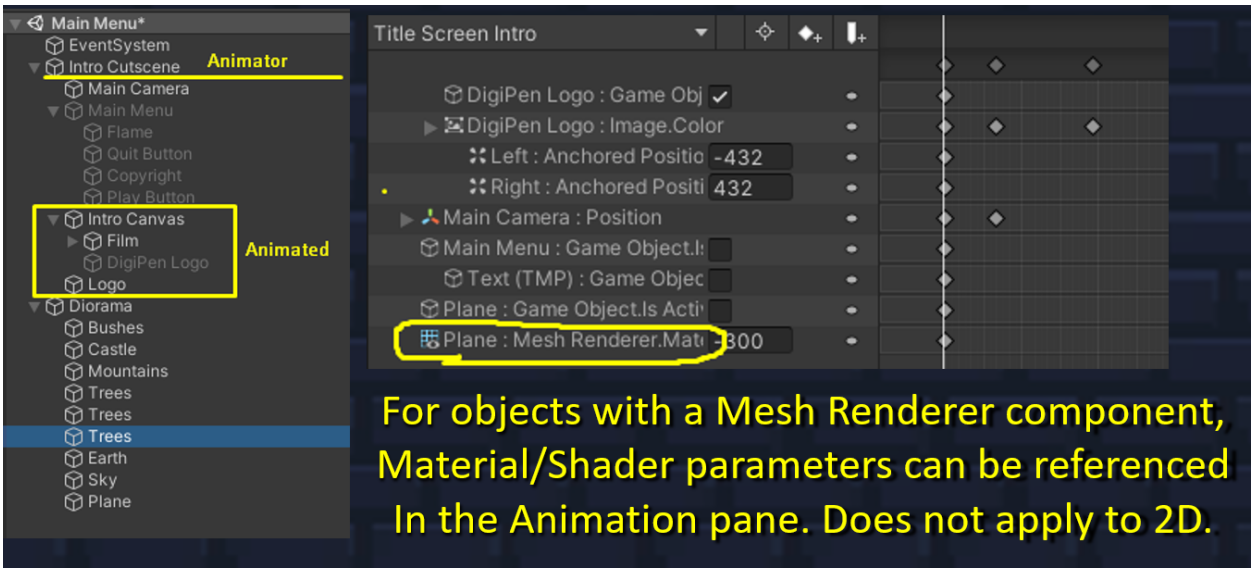
Custom lighting material implementation for 2D sprites and tilemaps; shown by the shadows and lighting interactions for the column (tilemapped) and sign (sprite).

Faux voxel effect provided by per-pixel extrusion given a texture projected onto a 3D plane.

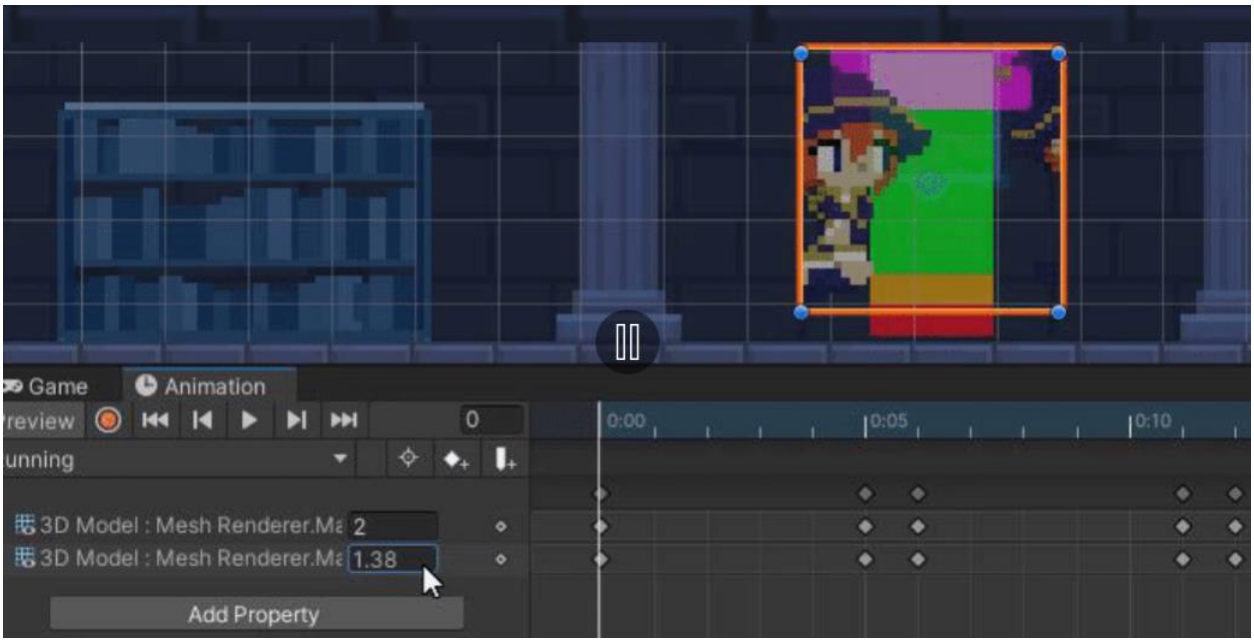
Additional layering of triplanar textures onto a material to allow effects like this; perfectly tileable cracks on a triplanar surface without the need of an independent texture.



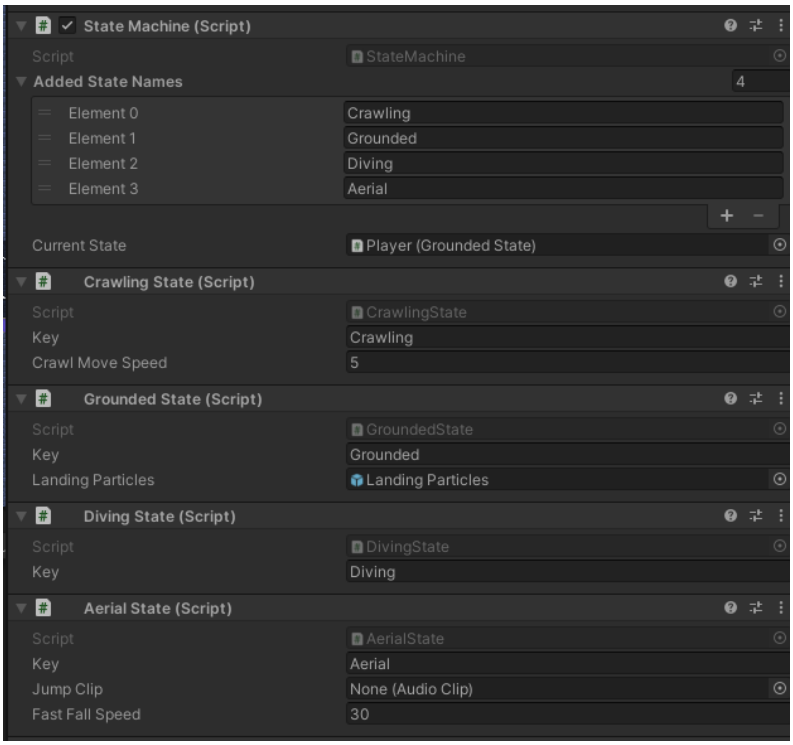
Further example of how the extrusion effect is used to create a faux diorama effect



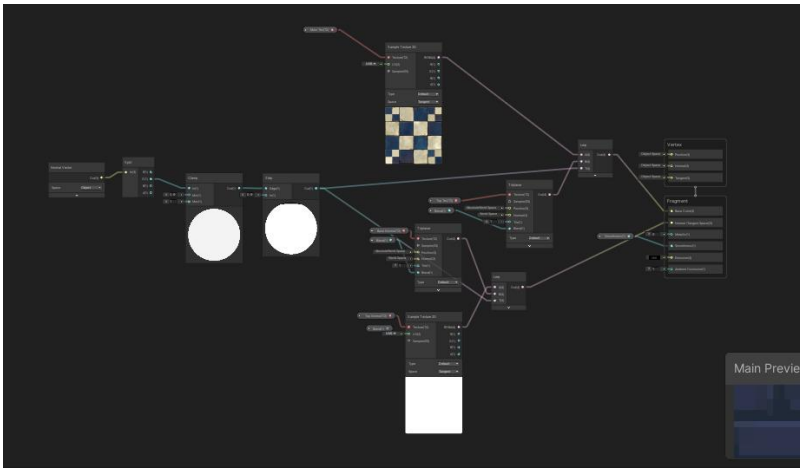
Exposure of URP shader properties to the animator for keyframing.



Exposing UV coordinates of currently displayed texture on a plane to move the cropped view of the texture to a specific coordinate/sprite on a grid.



Example of how specific player states are cut up into FSM base states, each able to adopt its own unique properties and parameters for specific state functions outside of what's provided by the state machine. This FSM follows a key/state dictionary in the machine, allowing states to easily be called and switched between using serializable instantiation.



Implementation Notes:

Everything is created using Unity and the base URP. No additional resources outside of the base Unity editor is required.