Project B: Flight Around a City

Evan Allan (eah8003)

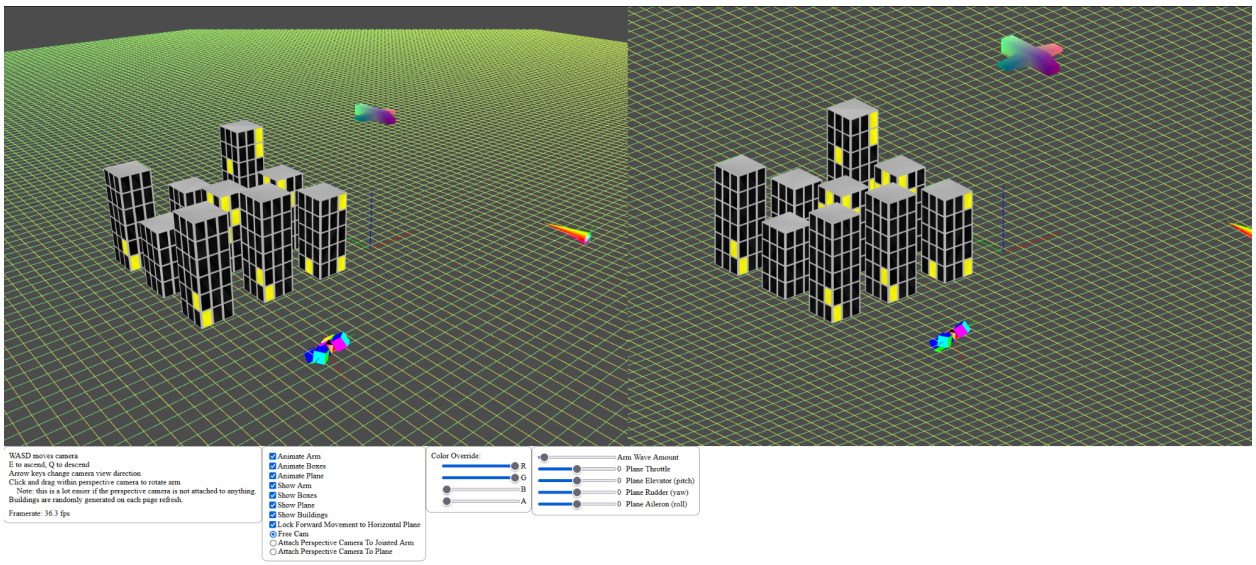
Goals

- Augment project A with additional objects
- Create a plane object with plane controls that the camera can attach to
- Attach camera to waving arm object
- Create randomly generated buildings which are different every time the program is run

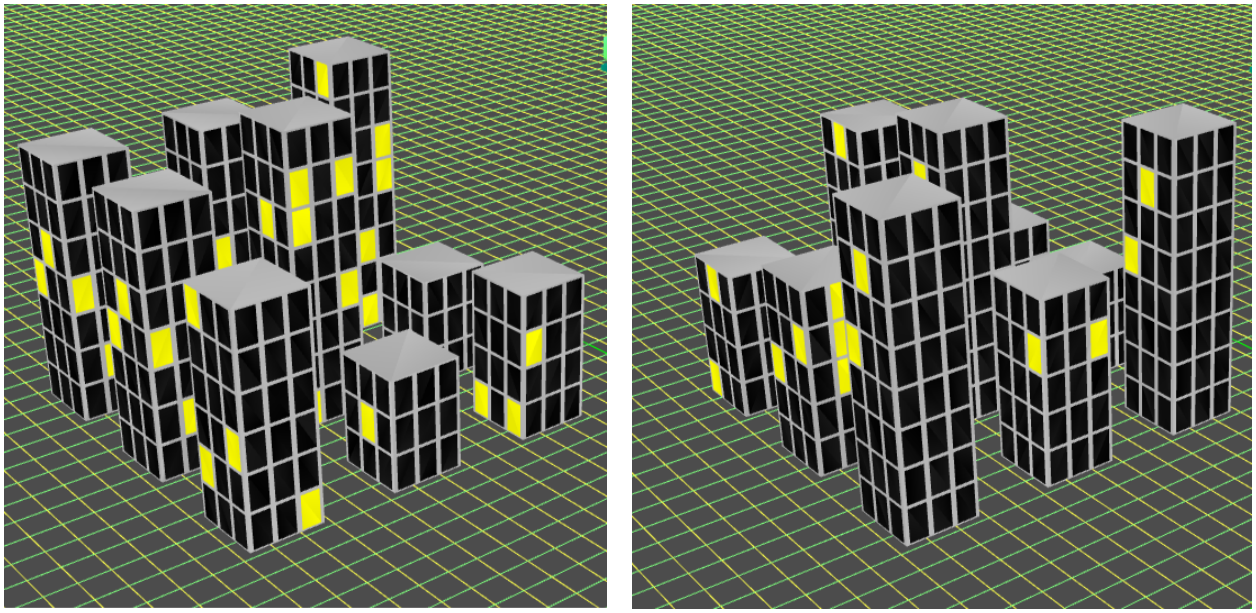
Controls

- WASD moves the camera
    - By default, this movement is locked vertically i.e. the Z position will not change
    - This behavior can be changed by unchecking “Lock Forward Movement to Horizontal Plane”
- Arrow keys change the view direction
- Clicking and dragging in the perspective camera rotates the waving arm.
- “Animate” checkboxes enable/disable animations.
- “Show” checkboxes show/hide objects on screen (note that animations will still happen in the background so when an object reappears it may not be where it was when it was hidden).
- “Arm Wave Amount” slider changes how wavy the arm is.
- “Color Override” sliders allow the user to override all vertex colors with a custom RGBA value. The selected alpha value will be used for blending with the original color.
- “Plane Throttle” controls how fast forward or backward the plane moves
- “Plane Elevator” controls the rate of change of the pitch of the plane
- “Plane Rudder” controls the rate of change of the yaw of the plane
- “Plane Aileron” controls the rate of change of the roll of the plane
- “Free Cam” allows the user to control the movement of the camera directly
- “Attach Perspective Camera to Jointed Arm” attaches the camera to the jointed arm such that it is viewing in the +X direction of the tip of the arm
- “Attach Perspective Camera to Plane” attaches the camera to the plane as if you were flying it

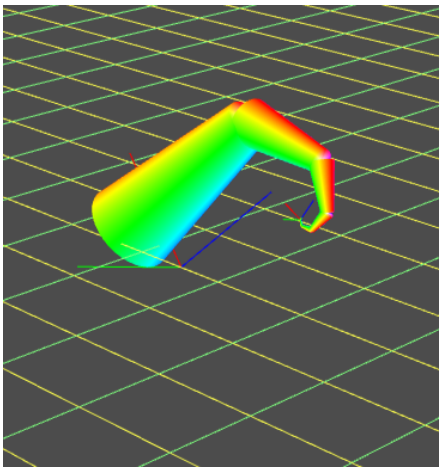
Results



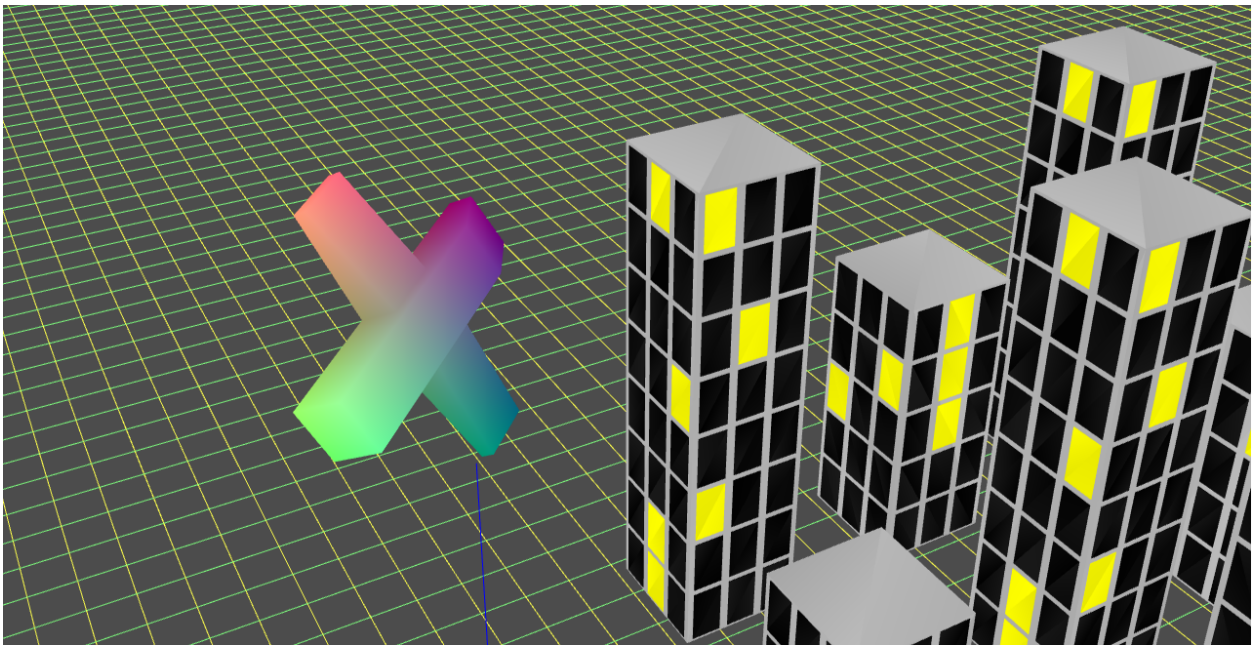
[Both perspective and orthographic cameras are displayed]



[Buildings are regenerated on each reload]



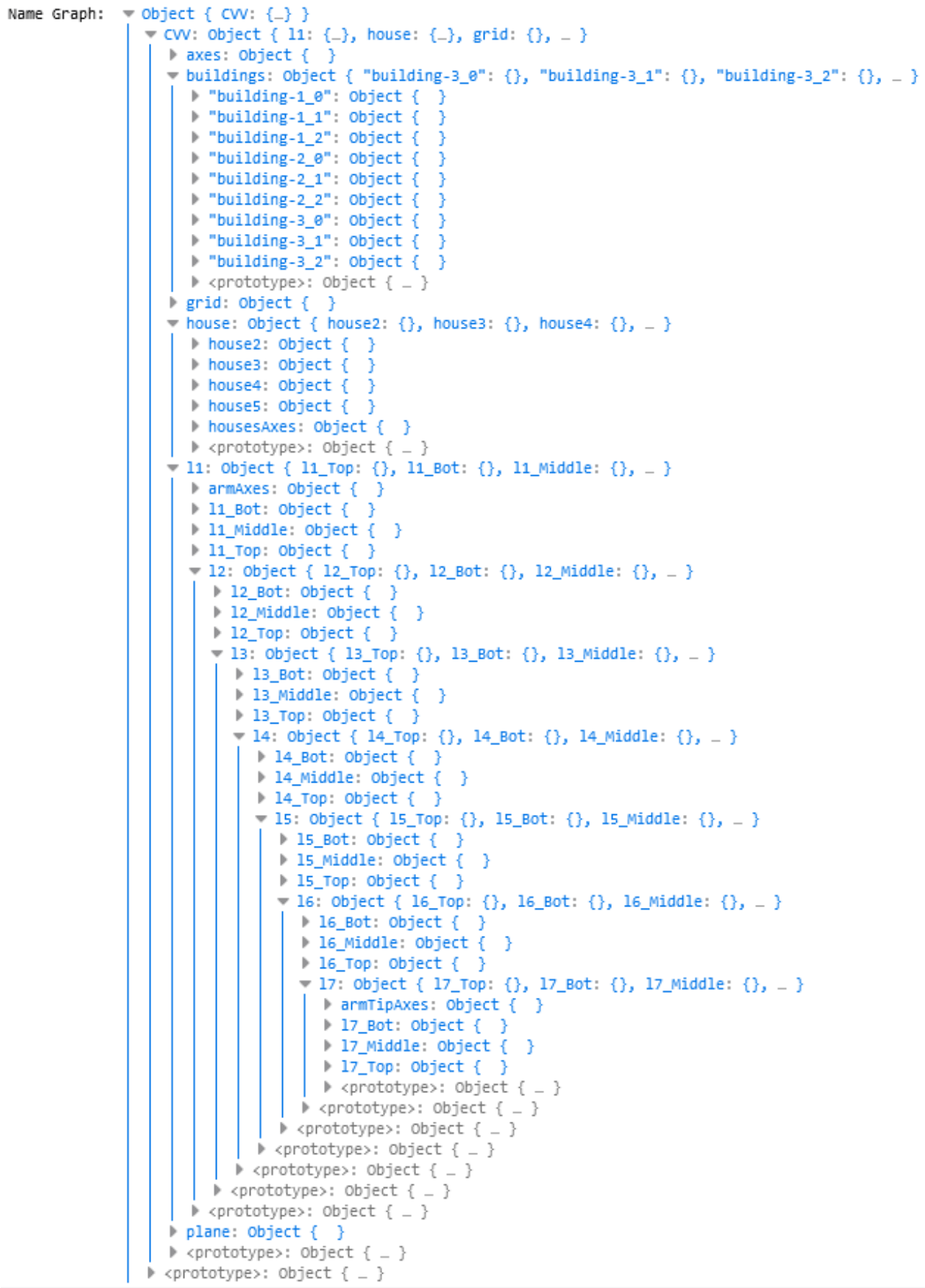
[Jointed arm can be rotated to any orientation]



[Plane can fly around the city]

Scene Graph

The scene graph is a bit large, as seen above. A bigger version exists at https://bit.ly/3Hm6ZNO. The graph creator I was using does not support the T shape for transforms, so I replaced it with a trapezoid with the shorter side on the bottom, which almost resembles the T shape. The squished cylinder object (for each of the sections of the waving arm) is split into multiple meshes - the circles at the top and bottom, and the curved rectangle around the middle (which is named “Cylinder” in the graph). This makes generating points easier. A single function generates transforms for the Top, Bottom, and Middle portions, so they will never be misaligned, and so they can be treated as a single part.
This graph was generated automatically based on the object hierarchy, which is shown below:


Name Graph:
Object { CVV: {…} }
CVV: Object { l1: {…}, house: {…}, grid: {}, … }
axes: Object { }
buildings: Object { "building-3_0": {}, "building-3_1": {}, "building-3_2": {}, … }
"building-1_0": Object { }
"building-1_1": Object { }
"building-1_2": Object { }
"building-2_0": Object { }
"building-2_1": Object { }
"building-2_2": Object { }
"building-3_0": Object { }
"building-3_1": Object { }
"building-3_2": Object { }
<prototype>: Object { … }
grid: Object { }
house: Object { house2: {}, house3: {}, house4: {}, … }
house2: Object { }
house3: Object { }
house4: Object { }
house5: Object { }
housesAxes: Object { }
<prototype>: Object { … }
l1: Object { l1_Top: {}, l1_Bot: {}, l1_Middle: {}, … }
armAxes: Object { }
l1_Bot: Object { }
l1_Middle: Object { }
l1_Top: Object { }
l2: Object { l2_Top: {}, l2_Bot: {}, l2_Middle: {}, … }
l2_Bot: Object { }
l2_Middle: Object { }
l2_Top: Object { }
l3: Object { l3_Top: {}, l3_Bot: {}, l3_Middle: {}, … }
l3_Bot: Object { }
l3_Middle: Object { }
l3_Top: Object { }
l4: Object { l4_Top: {}, l4_Bot: {}, l4_Middle: {}, … }
l4_Bot: Object { }
l4_Middle: Object { }
l4_Top: Object { }
l5: Object { l5_Top: {}, l5_Bot: {}, l5_Middle: {}, … }
l5_Bot: Object { }
l5_Middle: Object { }
l5_Top: Object { }
l6: Object { l6_Top: {}, l6_Bot: {}, l6_Middle: {}, … }
l6_Bot: Object { }
l6_Middle: Object { }
l6_Top: Object { }
l7: Object { l7_Top: {}, l7_Bot: {}, l7_Middle: {}, … }
armTipAxes: Object { }
l7_Bot: Object { }
l7_Middle: Object { }
l7_Top: Object { }
<prototype>: Object { … }
<prototype>: Object { … }
<prototype>: Object { … }
<prototype>: Object { … }
<prototype>: Object { … }
<prototype>: Object { … }
<prototype>: Object { … }
plane: Object { }
<prototype>: Object { … }
<prototype>: Object { … }