Names: Annette Chu and Carl Sun
NetIDs: alchu and carlsun
COS 426 Graphics
24 May, 2019

2D Floor Plan Builder

Introduction

Goal

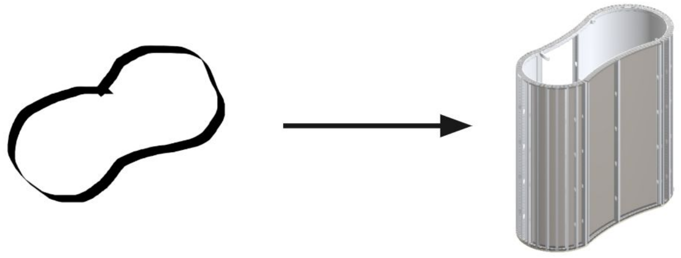
The goal of the project is to create a Computer-aided architectural design tool that allows the user to draw a quick floor plan and render the plan in 3D. Architects and designers would benefit from this tool when trying to quickly visualize how room sketches would realistically look. This tool is meant to cut out the time it takes to build a virtual model and take the user quickly from sketch to 3D model. The goal is to make it as intuitive and easy to use as possible with minimal learning curve.

Previous Work

Related tools that already exist include Rhino, AutoDesk AutoCAD, and RoomSketcher. Tools like Rhino and AutoDesk AutoCAD are heavy-duty computer assisted design tools. They take weeks to learn, and generating a model from a sketch can take an unnecessary amount of time if the goal is to purely visualize the room. However, both tools are successful in creating extremely realistic 3D representations. Given enough time, the user can model a floor plan to look exactly as they intended in a 3D space. Tools like RoomSketcher do not allow the user to see the room in 3D, rather they present a 2D representation of the room and allow a screenshot feature i.e. the user is shown pictures of what the room may look like. However, the intuitive floor plan drawing tool is extremely effective, as it limits the walls to straight lines and dynamically specifies dimensions for the user.

Approach

Our approach is a combination of both the heavy duty CAD tools and the intuitive RoomSketcher tool. We initially planned on showcasing a series of pre-set drawings and inputs. When given the input of a 2D drawing of a shape, the tool would render it into a 3D model of the shape raised in the z direction. We would accomplish this by relying on existing code from assignments, along with Javascript libraries such as Three.js. Some of our reach goals included being able to specify dimensions of models/proportions between objects, draw windows and doors into rooms, and having sliders to change parameters. We believe this will work well for architects who want to create a simple model with minimal effort to quickly visualize a design.



Methodology

We ended up allowing the user to free-hand sketch the floor plan and have the sketch extruded into a 3D shape. The position of the mouse was constantly updated and when the mouse was pressed down the positions at each timestep were stored in a spline array. This code was given in an example on the Three.js website, and our code builds on top of this. The spline array is passed into an extrude method when the mouse was lifted up to created the 3D space. This, however, created a solid shape that did not give a good representation of a room, as we were aiming for a hollow extrusion so that the user could zoom in through the walls. To fix this, we added a Path variable to the Holes array of our Shape variable used in the ExtrudeBufferGeometry method. This Path variable was calculated as follows:

- First, we calculated the centroid of the Shape that the user drew. This was done by summing the coordinates of all the points in the SplineArray, and dividing by the number of points. This gives us the center of mass for the Shape.
- Next, for each SplineArray point, we would calculate the vector from the point on the Shape to the centroid, and multiply the vector by a scalar of however thick we wanted our walls to be. This would give us a series of points that outline the hole, which we created as a Path variable and then pushed into the Holes array of the Shape variable before extrusion.

The extrusion automatically accounts for the holes when extruding. The shape, along with extrusion settings, were passed into ExtrudeBufferGeometry. The resulting geometry, along with the custom material we coded, are passed into a Mesh, which is then added to the scene. The camera was also set along the Z axis by 3 units.

The stretch goal we implemented was a GUI with sliders that control different parameters in the room: thickness of walls, area for floor plan, and height. When slider changes were made, the extrusion was removed from the scene, recalculated with the new values from the GUI, and rerendered. Currently, the height and wall thickness values work, but the area of the floor plan is buggy, as it stretches the extrusion out of the field of view for the user.

Helper functions were written to help with calculation. The function ThreeToTwo converts an array of Vector3s to Vector2s, in order to make finding the centroid and Path easier. centerOfPolygon finds the center of a given array of Vector2s by finding the average coordinate of the points. CalculateHole calculates the Path that the hole will take on.

Additionally, a global variable called drawingMode was used. We had encountered a problem with conflicting mouse events early on in the coding process, where the first time a user left-clicks, it both attempted to draw and rotate the camera. We wanted the left click to handle drawing first, and then rotating the camera after the extrusion occurs. To do this, drawingMode is a boolean that is initially set to true (because the user needs to draw first) and then turned to false upon the mouse button being released and the extrusion occuring. Any future left clicks will not pass the conditional statement that checks the state of drawingMode, preventing the user from drawing and extruding over the current extrusion.

Result

We measured success of our project by the extent we met our previously stated goals. We were able to roughly represent a room with the hollowed out extrude method, and change parameters to further specify the room's dimensions with the GUI we built. Aesthetically, the program could have made the room more realistic. However, we were successful in accomplishing the goal we stated: to build 2D sketches into 3D models quickly. With more work, this could be a very helpful tool for architects who own a drawing tablet.

Discussion

The approach we took is promising. One change we would make is to limit the user to drawing with straight lines and snapping to right angles (or more common used angles for ease of use). It would also be useful to have dimensions specified as the user was drawing. With more work, there are so many more features that can be accomplished with Three.js. Our future steps would include: being able to drag and drop furniture, drawing in doors and windows and have them built into the room in real time, specifying dimensions of models/proportions between objects, calculating square ft of material needed to realistically construct the room. Overall, we learned a lot about the process of graphics related software engineering while doing this final projects.

Conclusion

In conclusion, we accomplished our required goal of extruding any 2D floor plan into a 3D model of a room. We also accomplished our reach goal of providing a GUI that allows the user to alter some values of the room, and have the extrusion dynamically update.