# **Project Proposal** [30 pts] - complete

I plan to create a 3D engine from scratch with pygame as my graphics module.

* **Making things 3D**: since pygame only provides a 2d canvas, I need to render object that are stored in vertices and edges in 3D to a 2D plane. The perspective foreshortening effect would certainly be a challenge in the drawings. I plan to give the points a "depth" value to the vertices relative to the origin of the scene.
* **Color gradient**: since I would need to color the solids in a way that they appear 3D, I would need to shade the surfaces on the object. My plan is to make planes facing toward the positive x = y = z direction to be the brightest, which is also the color that is specified by the user. Then, I would calculate the normal vector for the planes and to see its angle with <1, 1, 1> to determine how grey and dark the surface should be. By "grey" I mean averaging the r, g, b value of the color and "dark" I mean proportionally decreasing the r, g, b value.

However, here is a challenge in this process that I would need to be able to determine the orientation of a surface. Assuming the normal vector for the surfaces are "positive" when they point towards the outside of the solid, I am still working on determining which normal vector to be the "positive" one when I only the points and the connection of the points, especially when they are custom inputs from the user.

* **Custom input**: I plan to allow users to input a list of points (they will need to enter each one in an indicated field, like Point 1: [entering field]) and a list of three points that form a plane, like Plane 1: [entering field]. I will need to use exception here to prevent the user from entering illegal things and give out warnings. The front end of this would be solved along with the foldable/collapsible panel. The parsing of the points and planes are relatively easy since it would be pretty much the same as the standard solids.
* **Rotation/moving**: the rotation and movement of the scene is actualized with a class Cam. It treats the objects as immobile and moves the window like a camera relative to the scene, (like Scroll for 2D). Some trigonometrical transformation would be needed for the rotation. The trickiest thing is to determine which planes to draw (only draw visible ones).
* **Shadowing**: casting shadow to the x, y-plane requires mathematical calculation for projection. Essentially, shadowing would be a grey projection of the solid to the x, y-plane. I am not planning to do shadowing on other objects.
* **Object overlapping and merging**: when two solids are overlapping with one another, I will determine how they would look.
* **Foldable/collapsible panel**: I plan to have a panel to get inputs from the user. However, since pygame does not have common UI options (button, drop-down menu), I will need to write it on my own or use outside module if I don't have time. This would be the most time-consuming work.

## Update 1:

* I will not be doing the object overlapping and merging since it is way too beyond my skill level.
* I will probably not do the shadow since it's also way too beyond my skill level. I will still try to tackle the color gradient though.

## Update 2:

* Changed the rotation modes to only along z-axis and slowed down
* Added axonometric view
* Changed the zoom method
* Added feature to draw a given function of z = f(x, y)
* Added labels to icon
* Added views of xy, yz, xz plane
* Show/hide grids

# Competitive Analysis [20 pts] - complete

Perform a competitive analysis (as we learned in class) on at least two other systems similar to the one you plan to develop in the file competitive\_analysis.txt, or .doc, or .pdf. If your idea is so unique that no other systems exist, find at least two systems that share a property with your planned project (for example, if you're developing a platformer game, find two other platformer games). Include at least five features in your feature table. In the summary, describe how the analysis influenced your ideas for the term project.

## Description of Your Planned Project

My project is a 3D engine that allows users to create a low poly style, 3D scene of solids made up with planar polygon surfaces. The program takes in instruction from the user to create built-in solids like prism (e.g. a rectangular prism or cuboid or box), pyramid (e.g. a tetrahedron), ellipsoid (e.g. a sphere), polyhedron (e.g. a dodecahedron, a solid with twelve faces of pentagons) with specified side lengths and color. User can also input a list of tuples of x, y, z coordinate for vertices and the list of tuples for a list of three points that form a plane. User can also change the position of the objects by changing the center coordinate of the object and rotate a certain angle about the center. I might implement more intuitive mouse drag to change the position if I have time. Users are able to move, zoom, and rotate the scene easily with mouse operations similar to those in Autodesk® Inventor with additional key operations to actualize more interactions. User can take screen shots of the scene.

## Evaluating Your Competition

#### 3D Graphics in Tkinter:

<https://www.youtube.com/watch?v=uuUbdluqSiE>

* a 3D graphics engine made by Peter Pak for his 15-112 Term Project Fall 14 in Tkinter. Some of the features include a Map Editor, FPS interface, Options Screen, and Cube Runner.
* The integration of 3D graphics and the shooting game was natural and good

#### Autodesk® Invertor:

* Although it is more of a 3D physics modelling and simulation software, there are some relevant features to my project. The mouse and key operation for its view feature is intuitive and inspiring. The 3D modeling feature is professional. The look of the surface is changeable and simulates the real material.

## Identify Comparison Dimensions

* Functionality: the more practical functions there are, the more likely users would be able to do what they want to do.
* Control simplicity: the more intuitive the operations are to the users, the more easily they could get used it and likely to keep user to use the application.
* Design: the design of the UI would be noticed at the first sight for the users.
* Smoothness: the application needs to run smoothly and minimizes lags.
* Entertainment: making users feel happy and rewarded is important to keep the users

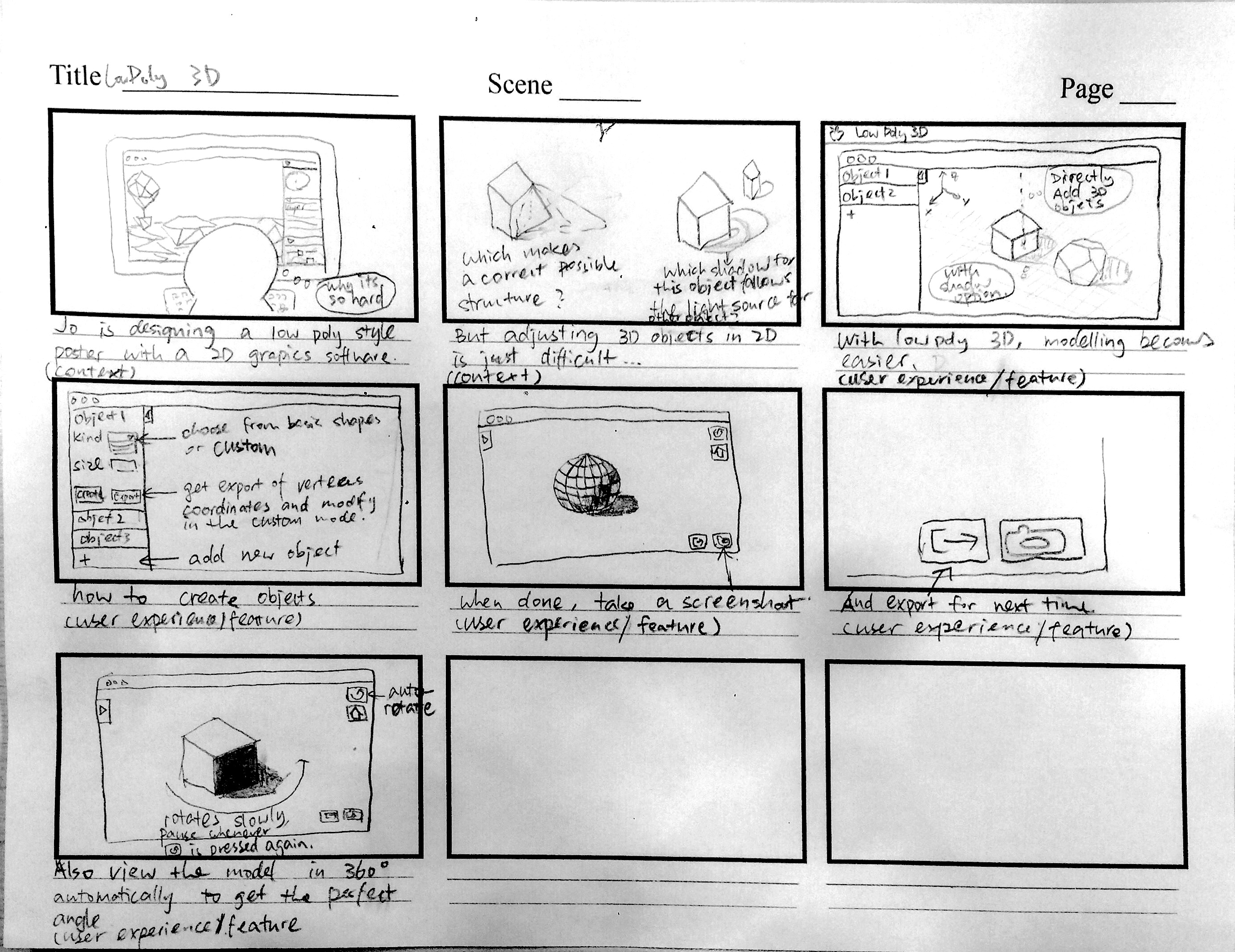
## Comparison Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Functionality** | **Control Simplicity** | **Design** | **Smoothness** | **Entertainment** |
| 3D Graphics in Tkinter | 2 | 2 | 2 | 1 | 1 |
| Autodesk® Invertor | 1 | 1 | 1 | 2 | 2 |

## Summary

* My project will be competitive in terms of UI design and the functionality. I have a very different design style from these two applications that it is more realistic than 3D Graphics in Tkinter but less so than Autodesk Inventor. However, being as realistic as Autodesk Inventor is not my plan anyway. I would like to take advantage of the 3D-ness look in low poly design but more abstract than real world objects. In terms of functionality, my main goal is different than these two so I can really use the cap to bring more creative thing to my user. Moreover, I add more mouse and key events necessary for my project to what Autodesk Inventor already has to make controlling easier for my users.

# Storyboard [20 pts] - complete



# Preliminary Code [20 pts] - complete

In addition to the design proposal, you should also submit code artifacts showing some early work on the project. This does not need to be a working demo; it should just be a real start towards solving the core problems of your project. For most students, 200-400 lines of decent code would indicate a good start.

# Timesheet [0 pts] - complete

In addition to the main deliverables, you should also submit a file named timesheet.txt. In this file, you should keep track of the time you spend on the project. You'll update this file each week.