Senior Comps Proposal

Bryanna Hernandez
bhernandez2@oxy.edu
Occidental College

1 Context

In stage productions and on-screen, a set designer is one of the unsung heroes of the visual medium. A set designer is responsible for bringing an environment to life while also embodying certain metaphorical themes like a character's journey. Using tools like scenery, the fixed backdrop of a stage, and set dressing, typically the decoration and furniture, set designers are responsible for modeling immersive scenes that are visually interesting and informative. Usually, a designer's creative vision is formed within their mind, but bridging this initial draft to the final product requires outlining, modeling, and refining a stage so it can be thoroughly communicated to the director, producers, and technical cast in charge of building the stage. For industry professionals, there are a variety of online tools that allow them to create in-depth and painstakingly detailed sets that productions can then spend thousands of dollars to realize.

However, for smaller community theatres and public school productions, these online tools are extremely complex, typically repurposed architectural programs that require hours of tutorials and practice just to learn which tools are relevant and useful to their purpose. Additionally, many of them are expensive and require money that could be more thriftily spent on the stage itself. Therefore, for my senior comprehensive project, I am proposing to build a small-scale set designing program that is appropriate for amateur and low-complexity productions. For my project, the main pursuit will be appealing to the lowest common denominator of needs in order to build a program that is useful but intuitive to use. My aim is to streamline the visualization process for these amateur set designers and create a final project that is simple and practical.

2 Prior Work

2.1 Industry Tools

The three primary tools used by industry professionals are SketchUp, AutoCAD, and Vectorworks. All three are self-described 2D drafting and 3D modeling software. SketchUp includes a "Push and Pull" functionality that allows users to extrude flat surfaces into 3D shapes. It also includes drawing layout functionality and surface rendering. Vectorworks allows paper and pencil sketches to be imported into Vectorworks using a similar Push/Pull command to create a basis for the project, and it uses a hybrid 2D/3D environment to see the direct impact of a 2D change on the 3D representation and vice versa. AutoCAD uses vector images and/or bitmaps and is designed for more architectural needs of plans and structures.

In terms of graphical user interfaces, all 3 applications have similar methods of dividing the screen into multiple sections, with a graphical representation in the center surrounded by toolbars. The SketchUp web-based application confronts first-time users with a toolbar on the left-hand side that allows users to start creating a model with a pencil tool to draw lines, an arc tool to make circles, and a square tool. Users can also import models that have been created by other users through an incorporated search engine. AutoCAD is divided into a Graphic Area, where users can make their designs; an Options Ribbon that has the most common actions; a pull-down menu bar with tool boxes; a Status bar that gives information about coordinates and grid control buttons in vector form; and a command line. Vectorworks has palettes, and small windows containing tools that allow you to draw and edit objects. Upon first usage, palettes are hidden so users can customize their space with the most useful tools. Palettes include "snapping" which allows you to snap objects onto a grid; "attributes" which decides the color, weight, and fill of an object or line; "basic" which is used for drawing; and "tool set" with 12 sub-sections for specific tasks like creating a staircase.

Both SketchUp and AutoCAD explicitly advertise the accessibility of user-created or outside assets that are made to be imported into other projects. SketchUp plugins from the 3D warehouse allow users to search for items as small as chess game pieces to entire buildings and even New York City. In AutoCAD, 2D Outside objects are known as blocks, representing furniture or other structures, and can be inserted into designs from its library.

All 3 software also include a way to export for a final design. In SketchUp, models can be exported in PNG or STL formats. AutoCAD's native file type is DWG, which contains information about the geometry of the file and is compatible with graphic design programs such as Adobe Illustrator and Corel Draw. Vectorworks' exportation is done

through PDF, and it is recommended that users export it to Photoshop to render textures and colors.

When breaking down these three primary applications into these categories, it is obvious that the biggest emphasis has been to create a workstation that allows users to create custom builds, draw or sculpt their own models from scratch, or allow access to outside user bases. For the sake of time, I plan to focus more on developing software for visualization purposes over from-scratch creation. By providing a standard library of pre-created props and allowing users to import props of their own, I hope to lessen the number of extraneous tools within my program, simplifying it down to its basest needs. Because my targeted audience has simpler needs, it will be less necessary to create highly detailed and customized designs, and the focus will be on creating an easy space for amateur designers to place and evaluate their scenes, switching back and forth between a 2D "build mode" and a 3D "display mode."

2.2 Building over Creating

When switching to tools that are more similar to my focus, I have highlighted The Sims 4 and Virtual Architect Home Design Software (VAHDS). The Sims 4 contains a build mode that can be used to edit a lot's architecture and construction while adding, moving, and manipulating objects on a grid system, the view of which can be toggled. It contains a build mode tool menu that can move, resize, and delete objects; a design tool that changes the object's color or pattern; and a "Save To My Library" tool among others. It includes a catalog with objects sorted based on rooms that it would appear in, including Kitchen, Bathroom, Living Room, and Study. It can also sort objects by type, including surfaces such as tables and counters and decorations such as paintings and posters. It also has catalogs for roofing, windows, doors, and columns. Objects are shown in a list as small 3D objects with text description upon mouse overlay, and dropping them onto the grid demonstrates the number of square spaces they take up, in a fully 3D visualization.

On the other hand, the VAHDS design system takes a top-down approach when it comes to adding walls and objects, choosing to specify measurements, angles, and lengths, rather than providing a grid system. The top of the interface contains a toolbar separated by Building, Interiors, Landscape, Terrain, Analyze, and Help. Within these categories, there are different subcategories of icons that will lead to a right-hand toolbar catalog, from which items can be dragged and dropped onto the screen, rendering in a simplified 2D drawing. A selected item will be displayed in the bottom right corner with its rendered 3D model in more detail. A bottom toolbar has more general tools like zooming in and out, rotation, and changing floor plans. Outside of its top-down view, cameras can be inserted inside the house with viewpoints on a swivel, so that users can see the 3D visualization.

Unlike VAHDS, The Sims 4 has 360 camera rotation capabilities, with various camera controls for viewing the environment. Using the keyboard or a mouse allows users to zoom in or out; directional keys allow the user to move the camera; the shortcut "T" goes into a top-down view; and using a mouse allows the player to pitch or tilt the camera. Because of these controls, it does not contain any automatic isometric projections.

The Sims 4 is a video game built for entertainment value, and its build mode was created in mind for designing full buildings. It can fully render textures and lighting. Its ability to completely render a 3D model is more preferable to some of the other industry tools that requires secondary tools. However, its camera layout when building the model leaves something to be desired. I prefer to take inspiration from VAHDS' fully two-dimensional build mode that transitions that user into a three-dimensional viewing mode. However, instead of making the user place their own cameras, I believe it would be best to pre-program up to 5 different viewpoints based on the stage's dimensions that allow the user to see their model from an isometric view, as well as sample audience perspectives such as in the front row or on a balcony. Taking this extraneous control away from the viewer with further simplify the software, allowing users to focus on the design rather than camera controls. Additionally, I prefer to defer to VAHDS' system of displaying props in a 3D render in the corner of the screen, allowing users to preview the objects before placement.

2.3 Timeline Tools

One last application I looked at was danceapp.us. While this tool is marketed for creating dancer formations, it has a wide range of applications in terms of blocking. Set designers are not necessarily expected to plan for stage directions within a script, it can be useful for the director and designer to collaborate and ensure their visions for the stage don't conflict before building the set. This application has a toolbar on the left of the screen that allows users to specify the number of people on stage as well as the dimensions. For example, three dancers on a floor that is sectioned in a three-by-three grid. Dancers can be renamed and given distinguishing colors. On the bottom of the screen is a timeline. Adding a new frame and repositioning the dancers makes them automatically move when the animation is created. It is a simple but useful application, and I believe that this basic method of blocking keeps the interface intuitive while fulfilling its purpose. Integration of this functionality will help set designers to keep the purpose of the scene in mind and imagine ways that their scenery and staging can facilitate the play.

3 Technical Background

3.1 Software Design

I am creating a brand new software which means transforming user needs into an intuitive interface as the software moves from a problem domain to a solution domain. Within this process there are multiple levels. The first, architectural design, is the most abstract version and identifies the different components that interact with each other within the system: the proposal for the solution domain. The majority of this proposal likely resides in this level.

The next level is high-level design which modularizes the system into multiple independent modules, dividing the software into its functional aspects and measuring each modules cohesion and coupling. Cohesion refers to a modules intra-dependibility, such as procedural cohesion, when elements within a module are executed sequentially, or communicational cohesion, which refers to elements of the module that work on the same data. Coupling refers to the inter-dependability between modules. If a module can directly access the content of another module, this is known as content coupling, while data coupling is when modules interact by passing data. However, a high level of independency is considered to be the best outcome.

Finally, there is user interface design which refers to the front-end of software development. UI provides a platform for the human-computer interaction, and it is divided into Command Line Interface (CLI) and Graphical User Interface (CUI). CLI is text-based and refers to a set of instructions. In order to allow non-programming users to interact with the software, a GUI system can be more effective. A GUI system includes the window, where the contents of the application are displayed; a menu, which is the array of standard commands grouped together in a visible place; an icon, a small picture represented an associated application; and a crsor for interactivity. It can also contain dialogue boxes, buttons, check-boxes, and sliders, but for simplicity, these may be beyond the necessary scope of the project. GUI design should typically be consistent, allow for short cuts, offer simple error handling, permit easy reversal, and make users initiators rather than responders to the system. Within GUI Design and implementation, it is important to have various tests for usability, compatability, and user acceptance.

3.2 Object Modeling

The main component of this project is related to computer graphics. Computer graphics includes object modeling, which is the computer rendering of geometric representation of objects. Rendering is the process of generating these images through light simulation, shading, color, texture, and related visual effects. It can also include animation, which is the changing of these objects that simulates an illusion of motion. For this project, I am not proposing the creation of a rendering engine from scratch, but it is still important to understand how it works so it can be properly manipulated. Rasterization is the process of projecting objects in a scene to an image plane. This means translating object surfaces, usually made up of multiple triangle meshes, usually in object order algorithms which are more efficient than image order algorithms which iterate over pixels.

Ray casting is the process of using geometric formulas to compute intersection, which can be useful for determing shadows or intersecting sightlines. Ray tracing simulates the bouncing of light paths caused by reflection and refraction, which requires various ray casting operations. Ray tracing can render effects like depth of field and blurriness, and it is most likely beyond the scope of this project. However, radiosity is an approach that breaks a scene into pieces and estimates the amount of light each piece gets, which is another method of rending shadows.

In this project, speed will likely be prioritized over extreme realism. Real-time rendering, often used for video game graphics uses rasterization but combines it with ray tracing and path tracing. In order to make it occur in real-time, it often relies on pre-rendered lighting for stationary objects, and it uses light probes for moving objects. However, rendering is usually a tradeoff between speed and realism.

3.3 Isometric Projections

When visualizing objects, there are different types of 3D projections. Orthographic projection is the easiest type because it ignores the z axis and project objects without considering depth. This type is not as useful for this project since a 3D stage necessarily implies depth. Another type is Perspective projection, which is similar to real world perception because it emphasizes vertices closer to the viewport by making them bigger. This projection is achieved through a perspective divide, by making the projected values of x and y inversely proportional to depth. The bigger the depth is, the smaller the resulting x and y are, so that things that are farther appear smaller.

Lastly, there is Isometric projection which visualizes 3D objects in two dimension, so that the x-axis, y-axis, and z-axis are equal to 120 degrees. This is a configuration of orthographic projection that ignores depth, and it is typically used to allow programmers and artists to create more elaborate environments. Isometric projections are often cheated by using 116.57 degrees for the x and y axes and 126.87 degrees for the z-axis, creating a 2:1 ratio that simplies trigonometric calculations and allows tiles to be 100x50, 600x300, or 64x32 evenly. This is a “pseudo” isometric

angle that seems useful for a viewport angle and allows users to see their models from a dimetric lens that simulates depth.

4 Methods

4.1 Codesign

One of the most important parts of my approach will be the codesign interviews. I am building this project for a very specific demographic, and I want to make sure that it meets their needs. Therefore, I plan to start this process by meeting up with a selection of amateur set designers who are active in the local theatre community to learn more about their process, their experiences, if any, with similar tools, and throughout the creation of the software, allow them to periodically review it and give feedback. For my first meeting, I do have a series of questions designed to start the exchange. First, I need to understand what, if any, experience interviewees have with the tools mentioned above in Prior Work. I also have some further questions to understand the user's experience with set design in general, so I have a question designed to understand how many productions and years the participant has been involved in as a theatre tech.

For participants who are inexperienced with set design tools, I want to understand more about their initial thought process when designing a set. I plan to guide participants through a brainstorming process, asking detailed questions to encourage the user to think deeply about a scenario they are unfamiliar with. Because this is an in-person interview setting, I will also ask the interviewee to draw as they are thinking. After this section, instead of asking leading questions about the features I have brainstormed, I want to ask questions about the participant's opinions regarding the problem the feature is attempting to address. For example, since one of the features is a prop library, I will ask about the participant's preferences regarding using super-specific props or more generic geometric shapes, and to brainstorm the top 3 props they believe would be used in their production. At the end of this section, I have included an open-ended question about any comments or additional points in their process they would like to add.

More experienced users will go through a similar brainstorming process, though for this section I would like to observe how they typically incorporate online software into their process, if at all. For software they have experience with, I will ask them to name their top three favorite tools and least favorite or useless tools so I can compare the programs' strengths and weaknesses and combat issues that need to be addressed more urgently. I will also include questions like "How long did it take you to learn this software?" to understand how intuitive the tool is. I hope to leverage questions such as these to create an application that avoids any of the mentioned pain points and can be a gateway for creatives interested in the medium.

4.2 Creating a basic interface

The crux of my project is creating an interface where an average user can place 2D models and props that will render into viewable 3D scenes. The first step in building this interface is choosing an appropriate Application Programming Interface (API) that will allow me to render 2D and 3D graphics. I have chosen OpenGL, a low-level modeling and rendering software package that has been used in prior work, such as AutoCAD, and works with the C++ language. OpenGL's Utility Toolkit (GLUT) with iostream allows a program to display a window with graphics and translates mouse and keyboard input into window output. This contains most of the tools required to build the front-end software since it will mostly rely on visualizations and allows only limited interaction in selecting and shifting objects on the stage. OpenGL also provides basic Graphical User Interfaces (GUI) libraries which will be necessary for any extra user abstraction in interacting with the software. The GUI library that I have selected is Crazy Eddie's GUI (CEGUI) which provides a layout and unicode-based font rendering, has integrated animation support, and is fully customizable. This will provide a suitable base for any necessary menus, allowing users access to simple functions like prop uploading, timeline animations animations, and camera changes.

Upon opening the software, users will automatically be presented with a standard, pre-built proscenium stage in a top-down grid view. From the left-hand side, they will see most of the menu icons, including saving, choosing from a prop catalog, inserting their own props, changing to any of the premade camera views, and adding additional lighting. The prop catalog will be a series of text descriptors, but clicking on the text will show a 3D rendering from an isometric view of the object, which likely have been drawn offscreen in a space far from the participant's stage perspective.

4.3 Specificities in Camera Views

As previously stated, the initial planning stages will be a top-down 2D view of the stage to ensure the user has an accurate perception of the stage's depth. At any point, the user will be able to transition to 3 of the premade camera views for the stage, including an isometric view from the right, a front-on view (center in the front row of the audience), and an isometric view from the left. OpenGL positions cameras as a vector in world space. Using its GLM mathematics library, I can create a view matrix that allows the user to

smoothly transition from these 3 points to clearly see the stage from multiple angles.

4.4 Prop Uploading

For this aspect, I plan to use an OpenGL OBJ loader library to transfer 3D props into the game. I will create a Model class that will read OBJ files and hold vertex and texture data. For all my props, I plan to pre-make them in Blendr to satisfy the OBJ file format. Because I haven't had very many interviews yet, I am unsure as to what types of props would be the best to make. I have been stuck between making very specific props which would include things like candles and books, more generic props like tables and benches, or very simple geometric props like simple spheres and cubes. I believe that I will probably end up making a mixture of all three and creating menu-specific categories for them, but narrowing down the specific props to make sure they are the most useful is still a work in progress. One of the aspects I want other users to be able to do is upload their own OBJ files, so it will be an option on the toolbar menu. By uploading an OBJ, the program will save the OBJ file in a certain database, that can then be later accessed in the prop catalog in a separate user-added category.

4.5 Timeline Animations

This feature will be separated from the rest of the menu on a bottom toolbar. How it works is that users will be able to place human prop models on a frame, effectively saving its position, and then will be able to move the prop and save its position on another frame. Using these two points, the object will be translated from one frame to another using a simple translation equation, dividing the amount that has to be walked with 24 frames in between the user-created ones. This feature is made to be basic and does not support complex animations. Humans will be shown to be gliding along the floor, and the users will not be able to alter the number of frames between the user-created ones to ensure simplicity since their main job does not revolve around actor blocking.

4.6 Lighting Rendering

For this section, my main objective is to include spotlights. For normal lighting, I will be using a directional light source which is modeled to be infinitely far away so it shines evenly on all objects. This will happen through creating a light Struc and defining its light direction vector so that it is the same for all objects in the scene. For spotlights, I will be transitioning to point lights, which the user will be able to add to the scene at will. Point light rays fade out over a distance, and this is known as attenuation. Using a light's position, we can calculate the distance the light show stretch and then its attenuation value to mimic a spotlight effect.

5 Evaluation Metrics

Because this is an app I will be building in codesign, it will be evaluated on user satisfaction. I will be asking the participants a series of questions to understand their experience: "How long did it take to create a basic scene?"; "How difficult was it to understand the interface?"; "Can they see themselves continuing to use this software in place of/alongside existing software?" The other type of evaluation will be benchmarks for my features. For most of my features, this will be an evaluation of their rendering in the scene to see if the models, lighting, and timeline animation work properly and create a useful visualization. For the prop catalog, I will want to include a wide variety of props, likely at least 15 simple props. Feature evaluation will be included in the user satisfaction surveys in order to get a better understanding of their usefulness to the participants and be able to make changes throughout the process.

6 Timeline

- 8/25: Semester starts
- 9/6: First codesign meetings have occurred, and a basic interface with a proscenium stage has been built
- 9/20: Premade camera views have been added and a Model class has been made with at least 2 props added
- 10/4: 2 more props have been added and a catalog interface has been created
- 10/11: Another round of codesign meetings have occurred about the most recent prototype
- 10/18: Changes and updates have been made according to the codesign meetings
- 10/25: User prop uploading features and timeline animations have been implemented
- 11/8: Lighting has been implemented
- 11/15: Poster due
- 11/22: UI and anything else has been cleaned up
- 11/29: Final round of codesign meetings
- 12/5: Poster presentations
- 12/11: Final round of ensuring everything is in its final state
- 12/15: Final paper and code due

7 References

Altieri, Alex. n.d. "Getting Started with Vectorworks Architect — What You

Need to Know." Blog.vectorworks.net. https://blog.vectorworks.net/getting-started-with-vectorworks-architect-what-you-need-to-know.

Gavin, Brady. 2018. "What Is Sketchup (and How Do I Use It)?" How-to Geek. How-To Geek. September 4, 2018. https://www.howtogeek.com/364232/what-is-sketchup/.

"Isometric Projection in Game Development." n.d. Pikuma.com. https://pikuma.com/blog/isometric-projection-in-games.

STACBOND. 2020. "What Is AutoCAD and What Is It For, Software for Architecture and Engineering." STACBOND. July 16, 2020. https://stacbond.com/en/what-is-autocad-and-what-is-it-for/.

"Vectorworks Basics — ArchiDabble — Architecture Resource Platform, CAD Packs, Portfolio Markups + More Services." n.d. ArchiDabble. Accessed May 10, 2024. https://www.archidabble.com/tutorials/vectorworks-basics.

"Virtual Architect Home Design Software — Virtual Architect." n.d. Www.homedesignsoftware.tv. Accessed May 10, 2024. https://www.homedesignsoftware.tv/how-to-videos/.

Vries, Joey de. 2019. "Learn OpenGL, Extensive Tutorial Resource for Learning Modern OpenGL." Learnopengl.com. 2019. https://learnopengl.com/.