Design concepts:

Interface Interaction:

There are several interface tasks that need to be performed. These are: Tool selection, Exporting/ Importing models, switching between modes or views and type selection for use with the painting tool.

Due to the small number of tools we can combine the tool selection and import/export functionality into a single wedge menu. This provides a 2DOF solution for selection of multiple objects while utilizing small movements to reduce fatigue. Additionally, it is fast to use as a single button press is required to bring up the menu and one small movement in the correct direction is used to select an object. It also allows for scaffolding such that experienced users can simply remember the direction of specific tools and choose not to visually display the menu. It does however, require a devoted button and in the case where there are many objects to choose from it may become unwieldly. However, it is a substantial improvement over a non-constrained menu or a menu that requires THI which could lead to fatigue.

Because of the same reasons discussed above it seems sensible to use a separate wedge menu for the type selection when using the painting tool. It is possible to use the same button to bring this menu up as the main menu depending on whether the user has selected the painting tool. Alternatively, it is possible to use a separate button. If the same button is used it will be necessary to have an escape option in the painting type menu to switch to another default tool (selector). Using the same menu design helps maintain consistency within the interface. User preference for using a single button vs two buttons to access the different menus can be A/B tested during the user-based design phase.

Since there are only two modes or views that the user can switch between and since this is likely to be a common task it makes sense to devote a single button to toggle modes. This will be discussed in greater detail in the navigation section.

Navigation:

As stated in the previous section there needs to be an effective way for the user to switch between First Person View(FPV) and God View(GV). One consideration that must be made is that this is not a symmetrical action. Switching from GV to FPV requires the user to pick a target location for themselves to ‘land’ at, however, switching from FPV to GV simply moves the user’s viewpoint up and backwards from their current location. However, as we will discuss in the Environment Interaction section selecting a target is a core functionality that the application must provide while in GV. Therefore, the user can simply use this tool to select a target to shift to FPV at. Once the target is selected the user can press the toggle button to switch to a FPV at that location. Once in FPV a second press of the same button will take the user back to GV by essentially moving the viewpoint upwards and backwards from the current state. The final consideration to make in this area is how to animate these changes of viewpoint. The two main options are a zoom in/out animation or a fade in and out while the viewpoint switches. Zooming has the advantage of allowing the user to maintain a certain level of orientation and knowledge of surroundings. However, this may lead to simulator sickness. These two options will need to be A/B tested during the user-based design phase.

Navigation or viewpoint manipulation will need to be implemented differently in each mode. In FPV we want to simulate the ‘player experience’ as much as possible and therefore plan to implement walking based travel. This has the added benefit of reducing the possibility of simulator sickness. Unfortunately, it is a somewhat slow method to travel over large distances and requires the terrain to be walkable. However, allowing the user to switch easily between modes allows for them to switch to GV to travel over large distances or past non-traversable terrain. Walking also has advantages over the other major options which are flying and direct teleportation. Flying can quite easily lead to simulator sickness and requires 3D movement which is often a significant cognitive burden for people and would require both joysticks. While we have some form of teleportation available via switching views direct teleportation from within FPV has 2 drawbacks. Firstly, it requires the user to be in line of sight to select their destination (as well as controls devoted to target selection) and secondly it can lead to user disorientation upon a sudden viewpoint change. Walking requires a single joystick for movement and avoids these drawbacks.

In GV we will use the world in hand metaphor as a technique of view point manipulation. The world will be directly mapped to the user’s non-dominant hand(NDH). This will allow the user to rotate and move the world relative to a static view point. Using the world-in-hand metaphor allows the user to position the world such that other interactions can take place from a comfortable position. This reduces user fatigue and allows for more natural gestures to be used for other forms of interaction. It is from GV that users will perform environment interaction. We will perform A/B testing to determine whether users prefer to turn off world manipulation during environment interaction or to continue to map the world’s position and orientation to the NDH controller.

* Location and Orientation aides
  + Landmarks
    - Need to be easily visible
    - Consistently shaped and labelled for orientation
    - Some method for placement of artefacts: Ray casting (either in FPV or God view)
  + Map
    - Mini-map
      * Very good for both placement and orientation
      * May take up too much real estate on HMD
    - Map overlay
      * requires devoted button
      * obscures environment for locating land marks
      * Very precise
  + Way finding arrow
    - Minimal
    - Not good for relative navigation
    - Needs way to set goal: Ray casting (either in FPV or God view)

Environment Interaction:

For environment interaction one of the principle functionalities of the application is the ability to select points and manipulate them across 3-Axis. There are two viable ways to select points, each corresponding to a different control technique.

The first selection option is ray-casting. Here the user simply points using a controller (Dominant Hand (DH)) to select a point. Since this means the virtual representation of the user’s hand is not necessarily close to the point it makes direct manipulation unnatural and difficult to implement consistently. We can therefore replicate a 3D widget representing the point in the virtual space near the user’s hand and manipulate that directly, mirroring changes to the actual point on the terrain. The downside to this is the extra layer of abstraction as well as the difficulty of catering for how sensitive the widget should be with regards to controller movements. However, there has been research in this area where the speed at which the user moves the controller corresponds to the sensitivity, i.e. fast movements lead to large changes while slow movements lead to small changes.

The second selection technique is the go-go hand method where the user’s hand is tracked non-linearly: the further it moves from the original position the larger the movements it makes in the virtual space. This allows the user to directly grasp points as a selection technique. Since the user’s virtual hand is always on the point to be manipulated in this case it makes more sense to incorporate direct manipulation of the point on the terrain. However, there is once again the issue of sensitivity of the controls, especially with distant points. Additionally, this control scheme often results in the user working with their arms fully or mostly extended which can lead to fatigue issues.

Both techniques have several advantages and disadvantages, it will likely take a basic implementation of both to establish whether either or both are practically applicable. If both are viable then further testing will be done during the user-based design phase. Both techniques also would be implemented using similar physical controls. The DH controller would be used for selection (either for pointing or reaching). Once a point was selected the DH controller would be directly mapped to the widget control axes (movement in the vertical and horizontal axis and rotation around the depth axis). Exceptionally, points along a user drawn curve (discussed below) have variables that differ on opposite sides of the curve creating 5 independent variables. In this case, we will utilise THI to allow for consistent direct mapping. The DH will remain continue to determine the vertical axis variable. The NDH and DH hands will manipulate the horizontal axis and rotational variables on the side of the line they correspond to, i.e. the left hand will control the variables on the left of the line from the user’s current view-point. It should be noted that in all cases movements that are not directly along a specific axis of movement or rotation will be ignored wrt editing environment variables.

It should be noted that while the techniques compete in this area ray-casting is more obviously superior for several other tasks mentioned below.

Other forms of environment interaction include painting on terrain types and drawing curves over the terrain on which points can be placed. These are, in fact, very similar and can both be implemented in a rather straightforward manner using ray-casting. This has the benefit over direct interaction of requiring far smaller movements thus reducing fatigue. The one drawback is that it may be hard to draw steady curves using ray-casting but this can be corrected using curve smoothing. With regards to the physical control requirements this can be implemented using the DH controller as a pointing device together with a button to indicate whether pen/brush is down. The implementation will be the same and the effect in the VE will be based on the tool selected.

* How to set points along the line during drawing?
* Toggle selection off