Work accomplished Sunday, 15-10-2

StreetView

Google StreetView has a Javascript API that you can use, if we're still interested in getting the panospheres from GSV.

Voice Recognition

- ◇ Google voice API represents a feasible option for implementing voice recognition in the search
- ◇ AT&T's voice recognition is also an option, but it seems like it'd be much more work to integrate it into a Unity project.

Two VR Modes

It seems to be as simple as just toggling `Cardboard.SDK.VRModeEnabled`in a script. N.B. that we should probably write some function that'd make it possible to navigate the panosphere by swiping, in the case that we're going to really do the two-view thing.

Build settings

Note that, in the build settings, the option `Player Settings...` needs several adjustments before building for the first time. Namely:

- ◇ Under `Resolution and Presentation`, the setting `Default Orientation` should be set to `Landscape Left`
- ◇ `Other Settings > Identification > Bundle Identifier` should get set to something that's representative of the build.

Making a skybox

It seems that making a skybox with a panosphere of the place that we want to display might be the best way to achieve the same effect that STS got initially. There's a fairly simple way to do this, once you've got the initial panosphere.

1. Import the image into Unity with the Shader option set to `Skybox > Cubemap`
2. In the Heirarchy tab, go to Main Camera and create a Skybox object, and set the `Texture Type` to `Cubemap`.
3. N.B. That I'm not sure how to apply this strategy of generating the environments to fetching the panospheres from StreetView, but it's also possible to offload that to an external machine/server.

Thoughts on menu and place selection in VR

One way that menus could be handled in VR is by building a scene in which the clicker on the side can be used to select the appropriate menu option, which can be found by raycasting through a certain reticule that the user lines up with the desired selection. These menu options could be represented as physical shapes on the screen, which might be good. Furthermore, this method of selection could be used if we're still going to do the virtual walk thing. The same method could be used with tapping, the raycasting would just be through an imaginary screen in front of the camera.

Drunk GTA V break!

Scene Switching

It seems like the best way to switch between scenes is with the function `Application.LoadLevel`. You pass that application either the index or the name of the *scene* that you want it to load, and it will despawn everything in the previous scene and then spawn the next scene. There is no difference between the "level" in LoadLevel and the "scene" in the regular unity editor. Yes, it is stupid, and yes, it is confusing.
We can have the nodes where we do the panospheres be all difference scenes. This also means that any kind of menu or whatever we have is going to need to be its own scene. Easy-peasy. The hard part is going to be writing the code to access whatever sort of framework that we want to use for storing and retrieving data.

In the project right now

In the scene pOlson - test from my fork of the Github Repo that we're working in, there's an example of a (slightly broken) panosphere. I'm thinking that the easier way to do it is going to be breaking it up into a cube, or using the software detailed in the devpost that STS made.
I've yet to work out the code for raycasting from the camera through a reticule, but it doesn't appear to be very difficult at all. Furthermore, I don't know how to trigger events in the event that the raycast does pass through an object; that'd *definitely* bear looking into, if possible.
I have several ideas for menu and UI design that I'd like to run by you guys, namely using the trigger on the side to bring up a menu while in the virtual panosphere.
I have a feeling that the searching for items around campus, like "where to buy eggs" and such things, is going to be what's really unique and special about our application. If that's the case, then it behooves us to not only populate our resources with buildings, but also potentially with basic things, like libraries and computer labs and what not. *Otherwise, we're essentially cloning what STS has already done.*