Project Journal - Team Green

Team members

Joshua Bender (Team Leader, Unity Manager)

Logan Stout (Version Control, Proxy Team Leader)

Canyn Black (Software Director)

Deekshitha Kallem (Documentarian)

Dingyi Kang (Asset Manager)

Lab Meetings

Thursday (7:00 PM - 9:00 PM)
Saturday (3:00 PM - 5:00 PM)

Project Summary

Main actions list:

- Car controls
- AI (city traffic)
- Pick assets to do this week
- Traffic lights stopping/driving
- Day and night cycle (with sun movement and clock in the car)

Assets list:

- Nine blocks
- Road with nature and lamp posts
- Main driving Car
- People walking around
- Traffic lights
- Fountain
- Park
- Other cars
- People
- trees

City Map:

- Center of the city - park with a fountain and running/walking track
- roads with traffic lights

Work Division

Joshua

Week 1
- Car interior (first-person view)
- Come up with objectives for the game (brainstorm)

Week 2
- Leftover car interior stuff
- Pull up a design plan with Deekshitha's help
- Work on steering wheel controls

Week 3
- One block of buildings
- Programming floater (help others)

Week 4
- Finish block 6
- Add nature to other blocks
- Fine-tune car controls
- Start working on Traffic lights

Week 5
- Optimize the project

Logan

Week 1
- Unity master scene setup (to be done by 10/15/2021) and create folders for each of us and create scenes
- Portals prototype

Week 2
- One block of buildings
- Portals working with basic unity

Week 3
- City border
- Day-night cycle

Week 4
- Finish city border
- Fix the lighting at night

Week 5
- Finish night lighting

Deekshitha

Week 1
- One block of buildings
- VR Input

Week 2
- One block of buildings
- Help Joshua with the steering wheel
- Scaling the buildings in the block

Week 3
- One block of buildings
- Research on Car dash UI

Week 4
- Get AI car and people models for Canyn and Dingyi to use
- Add car dash UI

Week 5
- Add extra details to blocks (nature, etc).

Canyn

Week 1
- The traffic system (halfway done)

Week 2
- POC for the traffic system
- Description of AI patterns for design documents
- One block of buildings

Week 3
- Traffic system
- One block of buildings

Week 4
- Finish car traffic
- Show Dingyi how to use

Week 5
- Finish traffic agents and implement in the main scene

Dingyi

Week 1
- ASAP mention the size of the block of the city so others can work on it
- Road
- one block of building

Week 2
- Car controls script
- One more block of buildings
- Scaling the roads

Week 3
- Car controls
- Finish roads
- Connect car movement to vr input

Week 4
- Gather and implement car sound effects
- Add people on the sidewalks using NavMesh
- Add missing sidewalks to blocks

Week 5
- Finish sound effects
- Add Pedestrian agents

Week 1 Journal

Joshua

(10/12)

Before the next meeting, I wanted to figure out how we could all work in Unity simultaneously without overwriting each other's progress in a Scene. I initially did some research into the specifics of multi-scene editing, since that dealt with most of the issues already. After a reading of the Unity documentation for multi-Scene editing, I decided to conduct an experiment with 4 scenes: one master scene and 3 other scenes that each contained a different GameObject.

One of the main issues with multi-scene editing is that you can't save or create any cross-scene references **in edit mode**. During runtime, these are allowed since Scenes can't be saved in Play mode. By default, you still aren't able to create these references by dragging and dropping in the Editor unless you change a flag in the EditorSceneManager class by script. However, you can still grab and create cross-Scene references by script.

Thus, I created a script containing a static class containing a single public and static generic function that can be used to obtain a GameObject (or one of its components) from a specific Scene using the name of the Scene and the path to the GameObject from the root of that Scene. I tested it using another script to obtain a Transform from an object in another scene to use as a source for a Parent Constraint. It works exactly how I want it to, and it's as easy to use as the GetComponent function.

Additionally, I also created a class for a ScriptableObject that stores the current multi-scene configuration and allows it to load all the scenes within it using a single click. It should make it quicker and easier to resume work after reopening Unity since you wouldn't need to manually reopen all the Scenes again.

There are a few things that we'll have to keep in mind when working with multi-scene editing that can't be avoided. People working within a Scene need to be mindful of not accidentally modifying and saving a different Scene. Using CTRL+S will save **all** modified open Scenes, not just the one being worked on. Additionally, (although less likely to be an issue), all Scenes we use must have unique names since the Scene is found by its name rather than its path. I can change it to use the path, but it would make it more cumbersome and tedious to obtain cross-references if Scene files are relocated. Similarly, if an object that is being obtained is at any point moved or renamed, the variables used for referring to that object need to be updated to reflect those changes as well.

Overall, I do think that using multi-scene editing is the way we should work on the project in Unity, but it may be wise to limit how many cross-references we use to avoid later confusion. When possible, all objects within a Scene should only need to refer to themselves, rather than objects outside of their regular scope.

(10/14)
I felt like there was a bit more I could do to make cross-references easier to use. I realized they would be extremely intuitive if I included a custom property drawer for them to be used by default inspectors. So, I worked for a couple of hours to create a script detailing a class for cross-references that held the Scene path and the object path as well as both a utility class and a custom property drawer. The utility class contains both of the functions I created a few days ago, modified to use the path of the Scene rather than its name. The property drawer was where most of my time today was spent coding.

I wanted to create popup selectors for both the Scene and the object being referenced. I had to first grab the list of open Scenes and format their paths to only display their names in the selection. I also had to handle several edge cases such as where the chosen Scene is no longer present, in which I append the name with “(Missing)” and add it to a separate list that the popup uses. There were too many of these cases to fully detail here, but some of these resulted from not restricting code from running during Play mode, incorrect order of operations, and the use of coroutines (which caused Unity to close unexpectedly).

I did eventually sort out (likely) all of the edge cases that would be problematic. I forced the popups to be disabled when entering Play mode since the Editor code can’t run during runtime.

I decided to use reflection to obtain the instance ID of the GameObject both during Edit mode and Play mode. This lets me find the object using its unique identifier. Now, if the GameObject is moved and reparented within its own Scene, all CrossReference instances will update to use its new path automatically. It also will find the object if it is moved to another Scene, granted that the Scene the search is performed on is updated manually. It now functions very closely to how I originally envisioned it, and I am mostly satisfied with it. There may be some lines of code that are redundant now that I may have missed during the enhancement. I don’t plan to go back to clean these up unless they become a performance issue since it doesn’t affect functionality.

(10/17)
Today, I began working on creating a car model from scratch in Blender that the player will be driving. I spent the entirety of today’s session sculpting the exterior shell of the car based on some helpful orthographic car projections I found online. I’m pretty happy with how it looks so far, although the mesh’s topology could be better.

Tomorrow I plan to finish the exterior by adding the side mirrors and begin working on the interior stuff. I don’t expect to have it entirely done by the time we meet again on Thursday, so I’m just aiming to have the entire mesh constructed by then and work on texturing and rigging next week, along with whatever else I’m assigned.

(10/18)
I decided to scrap the car I was working on and instead find a modeled one online to modify for our needs. I realized that modeling the interior would end up taking far too much time and wasn’t worth the hassle. I found a small low-poly car that has several parts separated and

moveable that seemed to work well, so I downloaded both it and its textures and set up the blender scene to work with more on Wednesday.

(10/21)
I haven't had as much time as I expected I would this week, so I didn't end up having time to work on the car. It won't be done before the meeting on Saturday, since that's the earliest time I could resume work on it. Instead, I spent a bit today brainstorming the ideas for various objectives and constructing a pretty substantial list we could pick from and add in depending on how much time we have.

The more obvious ones that immediately come to mind are delivery/drop-off missions where you would deliver packages, food, etc to various locations around the city. In these kinds of missions, we could take advantage of our navmesh to display a path to the destination either as a physical object in the gameworld (an arrow) or on the car's GPS. Additionally, we could set a time limit for the player to make the delivery to add some replayability for getting better times.

Of course, the natural extension of this type of mission is the one where you are driving people to various locations, rather than objects. A fun difference we could apply to these missions is a sort of sanity rating that decreases the more you break the law by doing things like running red lights or crashing into objects. You would then get a review when you drop off the person that is rated between 0 and 5 stars. This could also be influenced by the time taken as well to add some more difficulty.

From here I had to think a bit more abstract and game-like since the primary purpose of our application is of course to entertain. I came up with three ideas that still need to be fleshed out conceptually, but have some promise even though they aren't technically realistic.

The first idea I had was to have time trials spread about the city that requires the player to drive through a series of rings to get to the finish line. This would be somewhat similar to the infamous missions of Superman 64 if you want to visualize it. A way to possibly make these more interesting could be to also add obstacles on the path while the time trial is in progress so it's not just 90 degree turns the entire time.

This also gave me the idea for obstacle course missions in general. These could be less speed-oriented and more about simply reaching the goal. Obstacles could include moving lasers, ramps/jumps, walls, etc. Although interesting, I still don't know if this would be enough to differentiate them from time trials if those also include similar obstacles.

My final idea was for some sort of car chase mission. A few AI cars could chase the player around the city, which has roadblocks and barricades placed up randomly at junctions to stop the player from simply going in a straight line. The goal could be to simply avoid pursuing cars for a full minute. If we wanted to make it more interesting, we could allow the pursuing cars to wield weapons to shoot and damage the player's car as well.

Dingyi

(10/14)
Thought about the framework assets we need for building the virtual city.
Brainstormed with teammates about the potential issues of assets that will be used in this project
Figured out a way to relatively evenly distribute the assignment of the assets for each team member for the first phrase

(10/15)
Searched, downloaded, and tested a few building assets
Figured out the appropriate scale for each block of the city
Relatively evenly distributed the asset assignment for each team member
Wrote a file regarding the assignment of the assets, Provided an explanation for the assignment in the file.
Had a discussion with Joshua regarding the scale of each block of the city and figured out a reference object we will use for building our blocks of the city.
Worked with Joshua to figure out how to load different scenes into a single main scene and how to layout the scene of road and scenes of blocks into a single scene.

(10/16)
Built the framework for the city, namely one base scene for the road and nine scenes of blocks above the road scene.
Set up the main scene
Scaled and positioned the nine-block scenes relative to the main scene.
Updated the assets assignment file.

Deekshitha

(10/15)
Today I cloned our repo. I also had gone through the documentation of open XR in unity docs and tried to add VR input to the game. I think I have done everything right. I compared it to the first project and verified it too. But am still not sure. I will have to talk to Canyn and take his input on my changes or test it out in the lab and check if the VR input is appropriately added or not. I also pushed my changes into the git.

(10/21)
Added assets for a block in the city.

Logan
(10/19)

I imported the portal system and began testing to make it work with our project. After importing the necessary parts to make the portals functional, I spent around an hour switching between the team project and the provided portal demo, trying to figure out how to make it work. I was only partially successful in this endeavor. When looking at the portal, you can see out of the connected portal, as expected. However, it is currently impossible to actually teleport between the portals.

Canyn

This week I explored various ways to implement vehicle and humanoid AI via waypoints. I watched a number of videos and experimented with Unity. At first, I thought it would be fairly simple, but this has not turned out to be the case. There is not a lot to mention, except that I came up with and discarded a significant number of possible methodologies.

On October the 21st, Dr. Mayfield introduced and demonstrated Unity NavMeshes. After some deliberation, I adopted this methodology, though waypoints have still ended up playing a huge role in our solution.

Week 2 Journal

Joshua

(10/24)
Today, I worked on getting the car done and into Unity. I decided not to combine any meshes or materials since that causes its own issues and it isn't really necessary unless we experience performance problems later (which is quite possible honestly). There was a weird issue with the mesh for the car's frame when I was importing it in Unity for some reason. I decided to just fix its transform and save it as a prefab that we can use since that was easier and just as effective as a proper fix.

(10/25)
Today, I went ahead and created most of the documentation for the design document and handed it off to Deekshitha and Canyn so they can fill in their sections. I didn't really do much else today, but I should have enough time tomorrow to begin working on figuring out how Kallem and I should tackle user interactions with the car's steering wheel. I already have some ideas, but they aren't developed enough to warrant their documentation yet.

(10/26)
I began working on the steering wheel today. My idea was to take the point of the target (controller) and project it onto the same plane as the steering wheel. I would then use that point to calculate the angle between the wheel's forward axis and the point itself. While the wheel is held, the wheel is rotated to maintain that angle for every frame. When released the wheel would gradually move back towards 0 degrees.

This worked fine, but I wanted to prevent multiple rotations since we need to bind this angle to a -1.0 to 1.0 range for input. I created a complicated set of restrictions on how the angle could be manipulated, such as for allowing the angle to shrink (which is essentially sliding your hand across the wheel), but never allowing it to grow (since that would actually turn the wheel). This managed to cause gimbal lock whenever both points would overlap or directly oppose each other (when the angle is 180 degrees).

I couldn't fix it today, but tomorrow I plan to try adding a second point of reference that will also be used to track how the wheel rotates. This should allow me to use one point to recover the other from a gimbal-locked state and prevent such issues.

(10/27)
Continuing from where I left off yesterday, I decided it would be simpler and more efficient to rethink my approach to the main issue I was struggling with. During one of today's classes, I realized that I could just store the angle of the projected vector from the last frame, compare it to the angle of the current frame, and use the difference to determine which way the wheel was being turned.

I threw out the logic tree I grew yesterday and implemented this idea with much more ease than I had with that damn tree. It also just worked better as well. The logic behind it still depends on 4 propositions, but it's much easier to comprehend. In simpler terms, if the wheel is being turned and the current angle is less than the given threshold, then it is a valid rotation and the wheel is rotated up to -- but never past -- the threshold. Additionally, if the angle is currently equal to the threshold, the only valid rotations are those that move away from that threshold towards an angle of 0 degrees.

I'm very happy with how the wheel behaves now, so I spent the rest of my time polishing up how it is accessed and modified by external scripts, as well as how it looks and is used in the inspector. For scripts, I made it so setting a new target GameObject automatically resets the last known angle at the same time so that swapping hands is trivial later. For the inspector, I made a custom one that emulates the behavior of setting a new target, and I limited certain parameters to only be configurable during runtime either because they get reset during startup or because they shouldn't be enabled during startup.

(10/28)
Today I met with Deekshitha in the Lab to put together our scripts and get the steering wheel and VR player working. For the most part, everything worked immediately right off the bat once we figured out how to properly grab the wheel. There were a few issues with that specifically, most notably that the wheel kept unparenting from the car when it was grabbed because of how the script worked. I looked up a workaround online and instead used the grab script on a child object of the wheel that references the actual wheel.

There is also one issue I don't plan to fix regarding how I determine the neutral rotation of the wheel. I calculate a projected vector upwards a unit above the wheel. Because of this, if the car ever rotates on the X or Z axis, that position won't be correct anymore. In theory, this shouldn't affect gameplay since the car will only be driving on flat roads.

The only other thing I need to change with the script is to limit how far the target object can be from the wheel on the projected plane. I didn't do it during the lab, but it should be a one or two-line edit to the code.

Logan
(10/27)

I played around with the portal system again, to no avail. After being frustrated with making no progress, I instead decided to get a rough version of Block 2 finished and uploaded to GitHub.

Going back to the portals, it seems that when the player goes through the portal, it is correctly creating a copy of the player at the exit of the other portal. However, it is not moving the main camera to the perspective of the copied player model. After passing fully through the portal, the copied model is unloaded from the scene entirely. When walking backward through a portal you just went through, unity will sometimes rapidly change the position of the camera to create a flickering effect on the screen.

I think these issues are being caused by the input system, which the portal system uses. The original portal demo was built in an older version of unity before the new input system was installed. The obvious solution would be to switch back to the old input system, but I think XR System is built around the new input system, meaning XR and the portals are exclusive to one another.

The only potential solution I can think of would be to rebuild the portal system from the ground up using the new input system, and hoping that fixes the problem. However, this is only a potential solution and there is no guarantee it will work. At this point, I'm not sure if it's worth it to continue trying to implement the portals and find a new way of looping the scene, or scrapping the looping scene idea entirely.

Deekshitha

(10/26)
Today I worked on scaling the buildings to the proper size in alignment with other things in the city. I also added an input system to the project to handle the steering wheel.

(10/28)
Today I met with Joshua in the lab and helped him with setting up the steering wheel in the lab and we also tested the VR Setup.

Canyn

This was a week of abstraction. As our methodology revolved around the unification of disparate scenes, I naturally created a scene designed to match the main scene in scale which could serve as a dwelling place for the AI agents and corresponding NavMesh. This idea ended up being discarded, and my work was transferred into the main scene.

I did not seriously address pedestrian AI this week, as I realized it to be significantly less complex. I made a vehicle-shaped agent which was able to travel to a waypoint and detect when it had successfully reached its destination. The waypoint was discovered by the agent via Unity's search feature, though this method was ultimately scrapped in favor of a locally stored public variable.

I had already conceived the notion of linking waypoints as though they were elements in a linked list, though at this time it had not been implemented.

Dingyi

(10/21)
Downloaded polygon assets for the central park

Completed the central park which is block 5 and adjusted its position relative to the main scene

(10/23)
Added roads main scenes

Week 3 Journal

Joshua

(11/4)
Today I met in the lab with Dingyi to combine the car controls with our input script. It went pretty smoothly, although there were a few issues due to the hand meshes having colliders I wasn't aware of until near the end of the lab. Dingyi had to leave a bit early, but I continued working on the car for another hour or so. In that time, I redid the colliders for the player vehicle to surround (but not contain) it. I also adjusted some parameters for the steering wheel script, such as how far it's possible to hold the wheel from.

(11/7)
Today we had our meeting virtually since everyone was unable to go to the lab. We focused this meeting on making sure we had the project requirements done by the end of the week so we can do polish and extras next week. We're likely only going to have enough time to implement the delivery missions next week since we're more restricted on time than I thought.

I also spent a couple of hours after the meeting brainstorming with Canyn on how to handle traffic lights and intersections with the AI. We came up with a design that used a chain of waypoints and triggers to smoothly navigate the agent through turns. We also started implementing behaviors for having the cars avoid hitting one another or the player's vehicle. On Monday, I plan to work on the traffic lights after I make the progress report that's due on Tuesday.

Logan

(11/4)
I had a few minutes to kill this evening, so I created my second block by copying and rotating my first one. It's a bit cheap, but it'll work for now. I'd like to go back and actually work on making a new block if I have time. I also made a simple day/night cycle by creating a script that rotates a directional light, which can be adjusted through a speed modifier in unity. It took all of 5 minutes to make. However, it seems that there is currently no ambient lighting in the scene, so the city is pitch black at night. Right now I've resolved the problem by adding a static point light so the city will always have at least some illumination.

(11/5)
I had some extra time before class to get started working on the city border, so the player doesn't see out into the void. I just made a new scene, put a plane down, and started placing some buildings. I got around half of it done.

Deekshitha

This week I had to work on one more block of buildings and I researched how to add a speedometer to the car to reflect the speed of the vehicle we are driving around.

Canyn

Joshua and I devised a way to handle intersections and agent collisions. These are by far the most difficult parts of the task. Joshua took charge of making the stoplight system and I worked on creating smooth, controlled vehicle turning. We also came up with a way to prevent collisions between vehicles (including the players). This last was not implemented for some time though.

Dingyi

(10/27)
Resized roads in main scenes

(10/28)
Create car controller script using the keyboard as inputs

Week 4 Journal

Joshua

(11/9)
Today I fully scripted and animated traffic lights to work with our waypoint system. The implementation was pretty easy. I go through a provided list of stoplights and trigger them at a set interval in clockwise order. I also put together and submitted the progress report on Canvas.

Additionally, I started to work on fine-tuning the parameters of the driving script. This seems like it's going to take more time than the rest of my assignments this week. I capped the velocity of the car in the script so I could increase the max acceleration of the car since it was way too slow as is. A problem I'm still trying to work around is preventing the car from being able to flip during fast turns. I don't want it to turn super sharply, but I also don't want it to flip over. I think I'm going to mess with colliders during the lab on Thursday if I have time and use them to support the car from the ground.

(11/11)
During this morning's lab, I mostly focused on helping everyone with their objectives. Most of my attention went towards Canyn to help make sure he gets the traffic waypoints done. I also helped Deekshitha with getting the speedometer to work and be set in the car.

During my time at the lab this afternoon, I mostly just helped Canyn with the waypoints, since Dingyi didn't come like he said he would. Closer to the end of the night, Canyn also helped me with fine-tuning the parameters of the car. It's now very difficult to flip, and it automatically recovers after a few seconds when it does.

(11/14)
Today I took the roads and crossroads into Blender and turned them all into a single mesh. I also realigned them with the crossroads a bit better while I was at it and that's about it. I reinserted the merged mesh into Unity and tried to align it with the old roads as closely as possible. It's not pixel perfect, but the blocks themselves need to be scaled to match my adjustments.

Logan

(11/8)
Just some minor updates here. I finished the other half of the city border. Well. "Finished." It's just buildings, missing sidewalks, and vegetation. I think Josh was going to go around adding those things to everyone's blocks though. I also have a semi-fix for the weird lighting happening at night. I added some ambient lighting, which fixes the pitch-black building issue. I also decreased the intensity of our main rotating light, which makes the buildings being lit at night less noticeable, which still keeps things decently bright when combined with the ambient lighting. It's not a perfect solution, but it works in a pinch. I'll take another look at it sometime this week.

(11/11)
Added sidewalks to the city border. Took like 20 minutes, tops.

Deekshitha

This week I had to add nature to all the blocks. I added lamps, trees, plants, benches, and flowers to all the blocks except block 5 this week. I also found assets for cars and pedestrians for Canyn and Dingyi to add to their work.

Canyn

So much stuff. Intersection completed and perfected. Vehicles do not collide, although the way they avoid each other is by stopping abruptly. I made an intersection prefab for four-way intersections and began placing them at the relevant intersections in the city. Began the work of interconnecting waypoints (I think that was this week?).

Dingyi

(11/01)
Improved the car controller script and made the car move and rotate smoothly
Modified the car preface by adding the WheelCollider component

(11/03)
Finished tiling the roads and crossing of the whole city

(11/04)
Discussed with Joshua the design of Vive controller inputs
Joshua helped me to get the reference of inputs from the Vive controller
Updated car controller script which can work with the Vive controller inputs
Tested the player car

(11/06)
Created a block of the city

Week 5 Journal

Joshua
(11/17)
Today was the big lab day. I couldn't work for the first couple of hours because of a midterm I was working on that was due today at midnight, but I did offer advice and help to the others when they needed it.

We completed most of the remaining requirements today, aside from traffic since Canyn was sick. (He did work from home though!) The parts we collaboratively worked on were the sounds of the player and other vehicles, fixing the performance, making sure the project builds correctly, Redoing the pedestrians, adding details for the blocks, adding a button combination for resetting the player position, and implementing an objective!

The only things remaining to be completed are to connect all the waypoints for the traffic, add variations for the car agents, adjust the forward's friction of the car (maybe), adjust interactions between the car and the AI agents, and do the documentation. I hope to do all of these things tomorrow before we submit.

(11/18)
The final day of the project. At the lab today, I polished sound effects a bit more, and I worked with Logan to add sound effects for the delivery mission. I also added forward friction for the car that scales with its speed. Additionally, I found a value for Canyn to use for the mass of the AI car agents so they act less like paperweights when you hit them. I also created the user guide and cleaned up the project files once everyone was done making their changes.

I'm again going to leave my reflective thoughts here as I did for the last project. I think that the division of workload was certainly more equally diversified compared to the last project. Everyone had at least some significant scripting work to do, and the work of finding assets was also pretty divided, although it could have been more equally split still. How much work was assigned each week, although initially still too much, was also made more balanced and achievable towards the end of the project?

One thing that greatly helped us with working collaboratively was our decision to split the city up into multiple scenes that would be loaded at once. That allowed us to work on separate parts of the city and program at once with less of a chance of accidentally causing merge conflicts and overwriting others' progress. Although I made a script for referencing objects across Scenes at the beginning of the project, we didn't end up needing it since we had been able to self contain almost everything within the respective Scenes.

Once again, time was a very limiting factor for us. We initially wanted to do a ton more than we ended up being able to achieve. We were only able to implement one of my five objectives, and several quality-of-life features we wanted to add simply had to be shelved due to time constraints. Part of the reason that time became a limiting factor this time (aside from the scale of the project itself) was due to issues with certain requirements of the assignment taking multiple weeks to solve and fulfil such as scripting and configuring the traffic and pedestrian AI.

If we were to do this project again with the same amount of time, I would have the team put more focus on the functional aspects of the program first and foremost, rather than having people work on parts of the city simultaneously. This would likely have resulted in the functional aspect of the program being completed as soon as possible, and we then could have spent the rest of the time making the functional program look pretty, rather than spending our time making a pretty program functional.

Regardless, I’m satisfied with how our project turned out given the circumstances, and I’m happy with how the team performed and worked together in the end, (even if we had several merging problems at times).

Logan
(11/15)
Some minor updates to lighting. I changed the rotating light to be real-time, since we’re not baking any lights. I also made sure every scene was using the same lighting settings.

(11/17)
What a doozy of a day. A few hours before we had our 7-hour session in the lab, I modified the light rotation script so the intensity of the light would change as the sun would rise or set.

Once in the lab, Josh wrote up all the things that needed to be done for our project to be complete, and the work was split up between everyone. The first thing we focused on was optimization. We decided to try and build the game, to see if it would run better, so Josh and I made a script to load all the scenes on startup. It worked just fine, except that the main scene would load *every* scene, including another main scene, which would then load all the scenes. Including another main scene. It was a simple fix, we just changed the starting point of our for loop to exclude the main scene.

Next up on the list, we implemented a script that allows the player to teleport themselves into the seat of the car. All we did was when the player pressed the correct input, the script would find the X and Z distance from the player and the seat, and then add that to the position of the player, thereby teleporting them to the seat of the car. We also briefly tried to change the rotation of the player but scrapped it.

Next up, Josh asked me to change the way the brake noises worked, as the sound file we’re using has a brief beat of silence at the beginning, making it sound like the braking audio is delayed. After some digging around in the documentation, I found a way to adjust when in the clip the audio would start playing. So, using some simple if-else statements, I changed the start time of the brakes, depending on the speed of the car. I also made the brakes quieter, because they were stupidly loud.

I also went ahead and deleted the unused files relating to the portal system, before moving on to the objective system. Essentially, we have a number of Delivery Spots that are randomly determined based on an array of potential valid destinations. At the destination, a large, translucent Destination Marker appears, similar to the old N64 Spiderman pizza delivery minigame. Once you reach the Delivery Spot, you pick up a package, and a new Delivery Spot is chosen. Once you reach that, the package disappears from the car, and then a new Delivery Spot is chosen again. It's very rudimentary, but it works. I wish we would have more time to flesh it out.

(11/18)
Last day of work. Today is primarily just commenting on code and writing up this project journal. I believe Canyn still has some work to do, regarding the traffic lights. Later on, I'm going to try to add some sound effects to the delivery system. Once it's all said and done, we're going to go through the project files and delete anything we didn't end up using.

The sound effects were successfully added, but when implemented both the pickup and drop-off noises played at once, instead of alternating. I wasn't able to fix this in time before I had to leave the lab, but Josh managed to fix it while I was gone.

Canyn

Prefabs for three-way intersections and corners are now complete. I have connected all the intersections and corners. More car prefabs have been made, and I'm mainly working on arranging them nicely. The interconnection part took a very long time and was very tedious, but admittedly very rewarding.

Deekshitha

This week I reorganized the hierarchy of flowers in all the blocks as they were making the whole app crash because there were too many of them and we wanted to disable them if they are not in the range of the frame. But in the end, we decided to remove the flowers because they were not worth the effort we were putting in. So I disabled them in all the blocks for better performance.

Dingyi

(11/11)
Searched and added sounds of engine and braking of the car into assets
Wrote script related to above sounds effect of cars

(11/13)
Added sideways to block 3
Modified the script of sound effects of player car

(11/15)
Searched and added pedestrian character assets
Added nav mesh components for baking navmesh
Added associated scripts for baking and the pedestrian animation
Added and adjusted sidewalk in block 5
Adjusted the sidewalk of block 8
Experimented and made the pedestrian animations work in block 5
Baked nav mesh (which is on the top of the sidewalk) for all blocks

(11/16)
Added into each block enough amount of pedestrians (AI Agents) (about 40 for each block)
Tested the pedestrian animations for each block
Adjusted the positions of pedestrians so as to avoid traffic on the sidewalk for each block
Tested the pedestrian in the lab
Tested the car sound in the lab

(11/17)
Removed the sidewalk of block 5
Rescaled block 5
Deleted old pedestrians and added new pedestrians
Added new pedestrian models with appropriate amount into the game
Checked and adjusted the position of pedestrians to make sure they are stepping on the sidewalks instead of floating
Checked and adjusted the relative positions of navmesh agent and collider of each pedestrian
Added colors to the skin, shirt, pants and shoes of each pedestrian in the game with a distinct combination
Tested these pedestrians to make sure there is no traffic
Wrote a script to get access to all pedestrians in the game and make only 5 able to move around. But this script is not needed anymore. I tested this incomplete script in block 4.
Adjusted the positions of some pedestrians so that there are seemingly more pedestrians around the start position of the game
Tested and adjusted the positions of those lamp blocking pedestrians
Removed the pedestrianLoader script from pedestrian
Searched and added car driving sound sources into the project
Wrote script of sound effect of agent car
Test and adjusted the audio source of the sound effect of the agent car

Double checked and adjusted all lamps and trees around the sideway so that they won’t block pedestrians

(11/18)
Commented on all scripts written by me. Added reference of resources I imported into the reference document. Rebake nav mesh or almost all blocks. Formatted reference document.