Smart Parking - Indoor Navigation

Davi Seiji Kawai Santos
Ciências da Computação
Universidade Federal de São Paulo – UNIFESP
São José dos Campos, Brasil
davi.seiji@unifesp.br

Enzo Reis de Oliveira
Ciências da Computação
Universidade Federal de São Paulo – UNIFESP
São José dos Campos, Brasil
enzo.oliveira@unifesp.br

Pedro Henrique Masteguin
Ciências da Computação
Universidade Federal de São Paulo – UNIFESP
São José dos Campos, Brasil
pedro.masteguin@unifesp.br

Abstract

The rise of Virtual Reality in our lives is occurring more and more, being more present in every day of every one. The objective of this project is to develop an Augmented Reality application that would be used on parking lots. In this project, we will have an image, caught by a camera, and will make an overlay on it, showing the path to a selected parking space. This will help people to find an empty parking space or even their parking space on a condominium parking, for example. For this project we used Unity, a development platform that offers support to Virtual Reality.

I Introduction

Parking is a task that everyone who has a car needs to do everyday, and finding a parking space in a parking lot may be a problem, and could cause anxiety and stress [1]. In this project, we are working on a parking space with rotating spaces, so this application would help when the exchange of them occurs, or even to new residents to find out where his space is at.

This project is developed using augmented reality, which uses coordinates of the real world that are caught by cameras, which allows the connection between the real world and the virtual one [2]. AR is growing a lot, and in the future will be everywhere. It brings the possibility to help people's senses [3], in this project the vision for example, that with the help of augmented reality we will be able to see a path to the selected parking space.

We need to use the UNITY, an IDE that can be configured, to set an environment to develop virtual reality and augmented reality projects [4]. The use of this technology enables us to make a virtual parking lot, which has proportional measures as the real one that we want to do for the project.

So basically there will be an application that, when arriving at the parking lot, opens the camera and the user can select a space, and a path will be drawn there using augmented reality. With that people would save time finding spaces or in some occasions, will help them to find his own space in the condominium park lot.

II Related Work

After searching on the literature, we identified Unity as a highly suitable tool for developing Augmented Reality (AR) applications due to its extensive feature set and compatibility with multiple platforms [5]. Previous studies and projects, such as implementation of augmented reality application using unity engine deprived of prefab [6], have successfully used Unity for AR development, since it could integrate a good number of frameworks.

ARFoundation is one of these features that we can use on Unity, which is a framework built for Augmented Reality purposes [7], and it is possible to combine with other components, like ARkit for IOS. Using both of them permits an interaction with the device camera and the environment present in Unity [8], which will be a very important thing in our project. We will discuss these technologies in more detail later.

We identified several related works that pro-

vide valuable insights for our project. The first one is an indoor navigation on a building [9]. To do that was built a 3D model of the building on Unity, and using some AR tools, you could select a room on this build, and a path dynamically drawn on the device's camera. This work has some similarities to what we are going to do, since we have our environment, the parking lot, and we want to show a path to a selected place, a parking space.

Another related work focuses on the use of Augmented Reality in a parking lot environment [10]. In this study, AR technology is employed to dynamically identify available parking spaces as the camera scans the area, while our project uses AR to display a guided path from the user's current location to a selected parking spot that is already defined.

III Development

To develop this project, we divided it in four key phases:

- Prepare: planning and configuring the environment (Unity).
- Prototype: creating an initial model that has the main functions of the application.
- Tests: Testing the prototype so we can fix some errors and see what we can improve on it.
- Improve: refinements and enhancement to the project based on the test phase.

III.1 Preparation

First of all, we needed to carefully plan the project's scope and requirements. Firstly we decided what parking lot we were going to use for this project. We selected a condominium that has rotating parking spaces, so it would be very useful to find the new space that you need to park the car.

Following that it's time to configure the environment, in this case Unity. Initially, it was necessary to find the technology that we needed. The first one is 'AR Foundation' [10], a framework of Unity that helps the development of applications with Augmented Reality, and it is capable of combining with other tools like 'ARkit' [11].

'ARkit' is a tool that makes a connection between our AR application and IOS devices. So to run our project on an iphone, we need to integrate both technologies on the Unity environment. To do that we followed some instructions that can be found on an official website from Unity and explained how to configure it.

III.2 Prototype

With the preparation phase done, we proceeded to develop the prototype of the project. The prototype is an initial model of the application, that has its core functions, which in our project are:

- Open the camera.
- Select a parking space.
- Draw a path to the selected parking space.

The first step is to draw a floor plan of the parking space. To do that we needed to take the measurements there and replicate them with the correct proportions. The result is as follows:

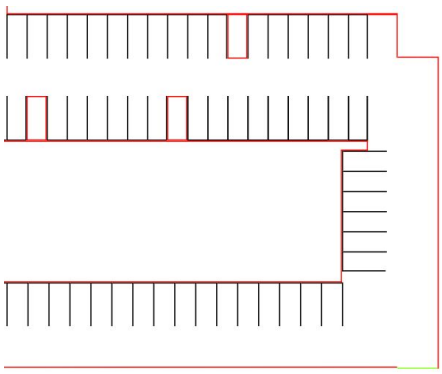

Figure 1: Initial design of the parking spaces.

In the next step, we put the image on the Unity, and began building the walls. These walls need to be in places where the user, who will be

in a car, cannot pass through. Therefore, our environment after building that walls becomes this way:

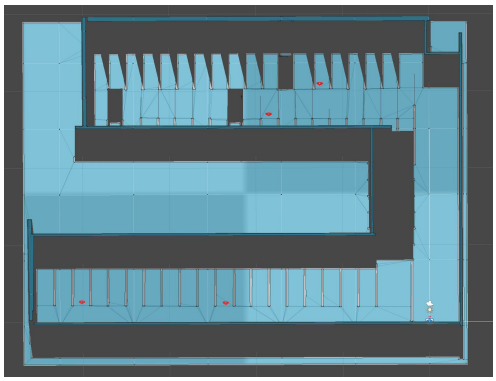

Figure 2: The parking lot, constructed in Unity, whit it's surrounding walls.

Next, we will implement NavMesh Baking, a feature in Unity that simplifies and automates agent navigation within a 3D environment. The NavMesh identifies navigable areas while accounting for physical barriers.

In our project, these barriers are the walls we've constructed, which represent areas the car cannot pass through. The NavMesh Baking process then determines and displays the most efficient path to the parking space selected.

After this, we can move on to the testing phase of our application. This stage will involve evaluating the system's functionality, ensuring that the pathfinding works correctly, and verifying that the user experience is good.

Last but not least, we placed the TargetCube on some parking spaces for testing purposes. The TargetCube acts as a marker that represents the goal for the car, indicating where it needs to go.

So finally, our project structure ends up in this way:

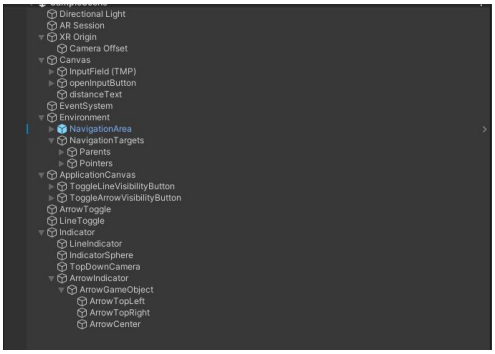

Figure 3: Project structure on Unity.

1. XR Origin: Contains the main camera, which is connected to the user's mobile device for augmented reality experiences.
2. Canvas:
   - Holds the central bottom button on the mobile screen.
   - Controls the visibility of the button and a text input field that appears when the button is clicked.
   - The input field allows the user to enter a parking spot number. Once the spot is entered and confirmed, the text input disappears, and the central button reappears.
   - Also includes the distance text that is updated by the navigation script.
3. Environment:
   - NavigationArea: Contains the structural elements like walls and the floor.
   - NavigationTargets: Holds the targets used to calculate the path.
   - Parents: Represents the actual targets where the path will be directed.
   - Pointers: The visual representation of the parking spots.
4. ApplicationCanvas:
   - Contains two side buttons:
     - ToggleLineVisibilityButton (bottom left corner): Toggles the visibility of the purple path line.

- ToggleArrowVisibilityButton (bottom right corner): Toggles the visibility of the arrow indicator.

- ArrowToggle and LineToggle: Objects that contain scripts to toggle the visibility of the arrow and line respectively (these objects are linked to the buttons).

5. Indicator:

   - The main object that contains the Control Navigation script, which connects the line and arrow navigation scripts.
   - LineIndicator: The child object that contains the settings and script for generating the line path for navigation.
   - IndicatorSphere: A sphere that represents the user moving through the environment.
   - ArrowIndicator: The parent object that contains the arrow navigation script.
   - ArrowGameObject: The child object that serves as a parent to three cubes, which together form a 3D arrow shape.

And the components that we use:

Figure 4: Components of the application.

1. ArrowToggle

   - Purpose: Manages the visibility of the arrow visualizer (the object that displays the navigation arrow).
   - Logic:
     - Initialization (Start): Checks the current visibility status of the arrow visualizer and stores it.
     - Toggle Function (ToggleArrow): Flips the visibility of the arrow visualizer each time the function is called, and updates the visual state accordingly. It also logs the current state for debugging purposes.

2. BounceEffect

   - Purpose: Creates a bouncing effect for an object, making it move up and down smoothly.
   - Logic:
     - Initialization (Start): Stores the initial position of the object.
     - Update Function (Update): Continuously adjusts the vertical position (Y-axis) of the object based on a sine wave function, creating a bouncing effect. The amplitude controls the height of the bounce, and the frequency controls the speed of the bounce.

3. InputManager

   - Purpose: Handles user input through a text field and manages the visibility of the input field and the associated button.
   - Logic:
     - Initialization (Start): Hides the input field initially and sets up listeners on the button and input field.
     - Open Input Field (OpenInputField): When the button is clicked, it hides the button, shows the input field, and prepares it for user input.
     - On End Edit (OnEndEdit): Captures the input when the user finishes typing and hides the input field, showing the button again.
     - Get Input Value (GetInputValue): Provides the stored input value when requested by other scripts.

4. LineToggle

   - Purpose: Manages the visibility of the line visualizer (the object that displays the navigation line).
   - Logic:
     - Initialization (Start): Checks the current visibility status of the line visualizer and stores it.
     - Toggle Function (ToggleLine): Flips the visibility of the line visualizer each time the function is called, and updates the visual state accordingly. It also logs the current state for debugging purposes.

5. NavigationController

   - Purpose: Calculates the navigation path to a target position using Unity's NavMesh system.
   - Logic:
     - Initialization (Start): Initializes a new NavMeshPath to store the calculated path.
     - Calculate Path (CalculatePath): Computes a path from the current position to the TargetPosition, storing the result in CalculatedPath. This function is called whenever the target position changes.

6. PathArrowVisualization

   - Purpose: Handles the positioning and orientation of the arrow that visualizes the navigation path.
   - Logic:
     - Update Function (Update): Retrieves the calculated path from the NavigationController. If the path is valid, it selects the next navigation point, positions the arrow, and orients it to point toward the next destination.
     - Add Offset to Path (AddOffsetToPath): Adjusts the Y-coordinate of each point in the path to align with the arrow's height.
     - Select Next Navigation Point (SelectNextNavigationPoint): Determines the next point along the path that the arrow should point to.

7. SetNavigationTarget (Line Script)

   - Purpose: Manages the navigation line and updates it based on user input, calculating the path to the selected navigation target.
   - Logic:
     - Initialization (Start): Sets up the line renderer and populates the list of navigation targets.
     - Update Function (Update): Listens for user input to select a navigation target. Once a target is selected, it updates the NavigationController with the target position, calculates the path, and draws the line using the LineRenderer.
     - Populate Target Lists (PopulateTargetLists): Automatically adds navigation targets and pointers to their respective lists.
     - Calculate Distance (CalculateDistanceToTarget): Computes the distance from the current position to the target and displays it on the UI.
     - Toggle Visibility (ToggleVisibility): Toggles the visibility of the line.

III.3 Testing

Upon reaching the testing phase, we moved to the parking lot we are using for the project and tested our application along with its core functionalities. This test will allow us to assess

how well our project is working and take note of any adjustments needed to finalize it.

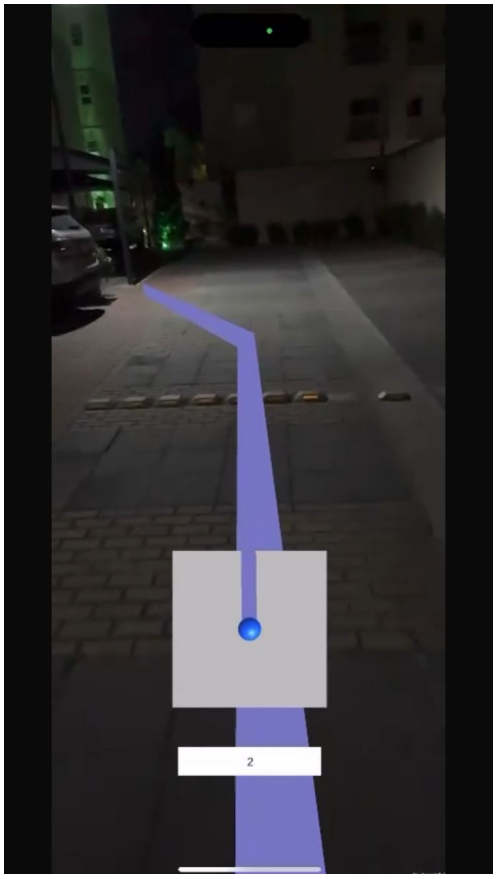

Figure 5: The application test.

The test was a success, with the application's core functionalities working perfectly. However, we noted a few areas for improvement in our project:

- Change the layout, giving options to the user to let the application show just what he wants.
- Make a button to show the menu, allowing users to select the parking space they wish to navigate to.
- Display the distance to the selected destination.
- Put a TargetCube on all parking spaces.

III.4 Improvements

The final step of our project is to enhance our prototype by implementing the improvements noted during the testing phase. This includes revising the layout to offer users more customization options, adding a menu button for easier selection of parking spaces, and integrating a distance indicator to show the proximity to the selected parking spot.

Additionally, we made these changes and made tests to assure that everything is going in the right way. Our goal is to have the app perform its core functions flawlessly while maintaining a clean, aesthetically pleasing, and customizable layout.

IV Results and Discussion

In response to feedback from the testing phase, we made significant improvements to the application's layout to enhance user experience. The final layout is shown below:

Figure 6: The application test.

The updated layout has these key functionalities:

- Button to toggle the blue line: Allows users to show or hide the navigation line, providing a clearer view of the path to the selected parking space.
- Button to toggle the red arrow: Allows users to show or hide the directional arrow that indicates the path to the target parking spot.
- Button to display the input field: Provides access to the input field where users can enter the desired parking space number.
- Distance to the selected parking space: Displays the distance from the current location to the chosen parking space, in meters.

Joining these changes with the core functions that were explained and made on the prototype phase, that are open the camera, select the parking space and draw a path to it, we have our final application. However, there are additional enhancements we have discussed that could further improve the application in the future. These potential improvements include:

- Integration with the condominium app - This would allow the application to automatically identify the user's designated parking space without manual selection, streamlining the process.
- Add more options to customizable the layout - Providing users with more flexibility to customize the layout, including changing the color and size of the navigation line and arrow. Additional options, such as switching to a dashed line or other visual styles, would further enhance user experience and accessibility.
- Build from anywhere - Developing functionality that allows users to initiate and build the navigation path from any location within the parking lot, rather than only from the start, maybe with QRcodes distributed across the parking lot.
- Additional Functionalities - Future enhancements could also explore the inclusion of real-time updates for parking space availability and integration with CarPlay.

V References

[1] Bajçinovci, Bujar, and Mejreme Bajçinovci. "Anxiety and Urban Stress for Parking Spots." Journal of Science, Humanities and Arts 6.1 (2019).

[2] Arena, Fabio, et al. "An overview of augmented reality." Computers 11.2 (2022): 28.

[3] Carmigniani, Julie, and Borko Furht. "Augmented reality: an overview." Handbook of augmented reality (2011): 3-46.

[4] Nguyen, Vinh T., and Tommy Dang. "Setting up virtual reality and augmented reality learning environment in unity." 2017 IEEE International symposium on mixed and augmented reality (ISMAR-Adjunct). IEEE, 2017.

[5] Simon, János. "Augmented Reality Application Development using Unity and Vuforia." Interdisciplinary Description of Complex Systems: INDECS 21.1 (2023): 69-77.

[6] Sivalaya, Gangavarapu, et al. "Implementation of augmented reality application using unity engine deprived of prefab." (2020).

[7] Chaudhry, Tanisha, Ash Juneja, and Shikha Rastogi. "AR foundation for augmented reality in unity." International Journal of Advances in Engineering and Management (IJAEM) 3.1 (2021): 662-667.

[8] Chaudhry, Tanisha, Ash Juneja, and Shikha Rastogi. "AR foundation for augmented reality in unity." International Journal of Advances in Engineering and Management (IJAEM) 3.1 (2021): 662-667.

[9] Nukarinen, Jesse. "Indoor navigation using unity augmented reality." (2023).

[10] DE INICIAÇÃO, PROGRAMA INSTITUCIONAL DE BOLSAS, and RELATÓRIO FINAL DE ATIVIDADES. "Localização de Vagas de Estacionamento em Imagens Capturadas por Câmeras de Celular." (2017).

[11] UNITY TECHNOLOGIES. AR Foundation. Disponível em: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html

[12] UNITY TECHNOLOGIES. ARKit XR Plugin. Disponível em: https://docs.unity3d.com/Packages/com.unity.xr.arkit@5.0/manual/index.html