


AR project with real-world-object interactions

Mousa Arraf

Patrisious Haddad




Table of contents

Introduction

In this project we are building an augmented reality multiplayer game that interacts in real-time with real-world objects, that can be utilized for in-game purposes as we are going to explain in further chapters.

The game consists of two players in an arena, both trying to kill the opponent by shooting at him. Each of the two players has a choice between a collection of four champions, both of the opponents have the same collection to choose from.

The interesting addition to the game we introduced is the real-world interactable objects, cube in our case, which is used to create a dynamic real-time changing arena since the cubes can be moved in the physical world to change the layout for the AR arena and change the battle conditions dynamically since each cube is a double edge sword for both its user and his enemy since it ricochets shots both ways.

Background Material and environments:

Unity –

Unity is a cross-platform game-building application that facilitates the process of game creation by providing users with abstractions and libraries so they can instantiate ready-to-use models right into their scenes with ease.
Unity also proved to have high standards when it comes to game graphics, when it comes to 3D spawned objects, they appear to be natural, with smooth movement and contribute greatly to the in-game experience.
One of the most useful aspects of unity we found was the "unity asset store", which makes game development even easier, and more cost-effective, in the asset store, you can find various models, textures, and animations.
On top of all of this unity has a strong active community, which as all programmers know is a big advantage to have, we can use the wisdom of the crowd to solve problems we face which others before us have already faced and solved.

Wikitude –

The Wikitude SDK is a tool for developers to create a more sophisticated AR model, the kit also serves developers that need image recognition and tracking technology.
Wikitude also allows spawning AR objects in a position relative to the image detected.
It is also worth noting that wikitude image tracking performance is way better optimized both performance and accuracy wise than that of ARCore,
So it was the clear winner between the two.

Photo from : https://www.wikitude.com/external/doc/expertedition/ImageTracking.html#multiple-targets




Simple image tracking in unity-

Using Wikitude's image tracker to a unity project is very simple, simply create a new game object and add the Wikitude image tracker script to it.
The image tracker itself needs a Wikitude target collecting which contains the information needed for the tracker to identify the tacked image, Which usually is features map for 2d images or cloud point for 3d objects, that is generated when the image/object is trained and the process searches for it in real-time when trying to track the image/object.
The Wikitude target collection (.wtc) can be created online using a free web-based tool. We can even create our markers inside unity itself.

Multiple targets -

We can also have multiple trackers simultaneously in the same scene with each tracking another collection. We made use of this feature since we needed to track more than one target, the first being the arena-spawning card, plus the champion's card and the cube card.

Wikitude SLAM –

SLAM short for Simultaneous Localization and Mapping, is a technology that uses feature points to understand the real world. This allows AR applications to understand and detect the physical world, and even detect 3D objects into the scenes. With SLAM we can use instant tracking: unlike image and object recognition which rely on pre-mapped targets to trigger the display of digitally augmented objects, instant tracking is markerless, so instead of requiring a target, it tracks the features of the physical environment to overlay AR objects.
Another feature offered by SLAM is object and scene recognition, which allows real-time $360°$ experience around physical objects. Scene tracking makes it possible to track feature-rich scenes at a larger scale that is defined by marker mapping.
Although this feature offered by Wikitude seems to meet our needs, practically it did not, since we are dealing with a dynamically changing scene by definition, so it was hard to keep the augmented reality arena in place and ensure a smooth game with so our scene changing constantly, which is due to the fact that we have a physical object that we keep moving so in turn it could overlap some of the scene feature points(which slam is trying to use) causing the game to lose track of the arena.

Photon –

Photon Unity Networking (PUN) is a unity multiplayer game package, it allows distant users to join each other in rooms where spawned objects can be synced over the network.

PUN's structure-

If we get to know the structure of PUN, we can understand why this package offers such valuable abstraction to the whole sterilization process.

- At the highest level, the PUN code implements unity-specific features like networked objects, RPCs, and so on.




- The second level contains the logic to work with photon servers, like callback functions. This can be used independently already (also called real-time API).
- At the lowest level we can find DDL files, which contain serialization protocols.

After we have taken a look at the structure encapsulated in the PUN API, we can see why this abstraction is so useful: it connects users to synchronized rooms without the need for the developer to get into the synchronization of the objects.
But to provide an even smoother experience we even manually edited the synchronization script to further enhance the synchronization performance, to match our needs without sending redundant packets over the network.

Gameplay:

Components –

the game consists of two adversary players that participate in a battle that takes place in the augmented world in front of the players. Each player must have an initial card to specify the location of the arena, that point then becomes the center point of the playing field. For the player to take place in a battle he must also choose a champion, each champion is represented by a different card, the player can scan to choose that champion during champion selection phase, then the player will be able to use the previously scanned champion in the battle arena, during the next stage-battle phase.

Objective –

Each player participates in battles over the network with one other adversary, both with their champions and shield cube. Each champion has health points that he can lose as a result of a collision with a fireball shot by the opponent, upon losing his entire health points he dies, and the champion's owner loses. The player can help his champion in the playing field by shielding him from incoming fireballs by moving his shield to fend the fireballs coming from the opponent's direction of. The shield block is a double-edged sword, in addition to protecting the champion, it can also interfere with the fireballs shot by his champion which will lead to the fireball ricocheting in his direction if not placed carefully.

Login –

At launch the game, a textual field appears where the user is asked to insert his username. This username is designed to distinguish the user inside the game. This username is displayed near the champion the player chooses to battle with.

Pre-game phase –

After the user signs in with a username, he can choose a champion to battle with in the arena, the game asks him to scan a champion card as we explained earlier in this document, the player has a choice between 4 dinosaurs, when the marker is recognized (the marker being the champions' cards), an AR dinosaur appears on top of the card, if the user chooses to select scanned dinosaur he can press choose, then the champion is selected. Then the player is asked to scan the arena card, the arena




card sets the place the arena will appear on, the player can adjust the arena card to place the arena on the playing surface, he then can press on Place to pace the arena in place. The last card he is asked to scan before the battle begins is the cube card; the cube card should be scanned in a way that the marker – a bear-like figure – facing the camera. The cube should be placed in a place that appears on the arena so that the user can use it in the battle. The arena card and the cube should be placed on the same level so that the cube appears on the arena and can bounce the fireballs off of it.

Game queue –

Once all the cards are set in place, and the champion is selected, the user enters the room selection phase, if no open rooms exist (he is the first user in the system, or all the other rooms are full) , we create a new room and admit the player to it to wait for another user to connect. When another user is found, he is then admitted to the room, both of the players see the other player's position in real-time through their phone screen, both positioned on the arena, and see the other player's cube as AR, but see their own cube in their phone screen just like the physical one.

Battle time –

After setting up the game and finding a match, both of the players are ready to battle.
Throughout the game the players can :

- Move to the right and left to dodge the incoming fireballs fired by the adversary.
- Move their own physical cube on the battle arena to their advantage, as they see fit.

Each player's purpose is to avoid being hit by the incoming fireballs while trying to hit their rival. Since the arena changes dynamically and since the fireballs bounce in different directions, the player should adjust his playing strategy to the current situation. On each hit the champion's HP is deducted, once the champion's HP reaches zero, he dies and respawns after 5 seconds to play again, against the same rival if he chooses to stay in the same room.

Augmented Reality :

A significant part in the game creation was ensuring a smooth AR experience, starting from the position of the arena, spawning cubes on the arena, and interacting with them based on their physical location … for that purpose, we tried different solutions to detect and interact with the real world in the AR arena.

Wikitude SLAM vs image tracker –

Placing the arena and ensuring it stays relatively stable is crucial for the game experience, for that we tried two solutions by Wikitude: SLAM and image tracker.
As we explained in a previous section, SLAM uses physical-world feature points to map out the field and place AR objects on top of it, it also allows rotation of 360, something that can be perfect for showroom purposes, but in our case, we create a dynamically changing field which can conflict with the technology used in SLAM.




Another method that proved itself was using Wikitude's image tracker: this method uses pre-defined targets and detects their existence, position, and orientation in the camera scene, we used a marker to set the location of the arena, the location of the arena is determined according to the middle-of-the-arena tracker, thus, by keeping the center of the arena marker in the frame, we will see the arena in the same place, hence achieving a smooth stable arena, that we can place on any surface we see suit. And by spawning our arena, next to the image tracker instead of on it, we allow the moving of our physical object(the cube) without interfering with the image tracker itself or taking it out of frame/focus.

Interacting with the cube –

As we mentioned in a previous section, each player has a cube that he can place on the playing field, this block can bounce the fireballs shot by the champions off of it, so it is important to detect its place accurately and transfer it into the playing field. To achieve the said purpose, we used an image tracker, since it was very reliable, on one side of the cube, and attached an AR cube on the physical cube with the same pre-defined dimensions. The attached AR cube is transparent to the user who owns the cube, so that he sees the actual cube on the playing field, but his adversary sees the AR cube in a solid color, thus creating an AR cube on the other player's side which allows him to know the location of his opponent cube, and vice versa. Using PUN that we discussed earlier we transferred the cube's location and orientation over the network, thus creating an arena with two interactable objects that bounce fireballs off of them. As we explained earlier, image tracking also allows us to detect the direction and the depth of the marker, by using said attributes we can introduce a cube that can be placed not only in parallel to the champion, and the shots reflect off the cubes depending on the directions they are positioned.
As a result of these cubes being moved by the players in real-time, we achieved an arena that changes dynamically which can change the playing strategy and be a real obstacle.
Creating endless variations for the battle arena that would need the player to adjust him game style accordingly.

Real-Time battle –

As we have discussed earlier, we used the infrastructure offered by Photon (PUN) to create an online game, but since we can't entirely rely on the network to work with no latency(since we have to account for packet travel time over the network) and no packet-losses we have put a system in place to try and minimize these effects.
In the PUN network we did not only synchronize the objects' location, but we also sent the velocity of the player; we did to compensate for the time it took for the packet to travel, in which the player have changed his location. The receiving end can make use of the other player's velocity at the time the packet was sent and calculate based on the time it took for the packet to arrive at a more precise location of his opponent at the given time.
Which in short would mean that we update the player position using opponent.sent.position + opponent.sent.velocity * packet.travel.time



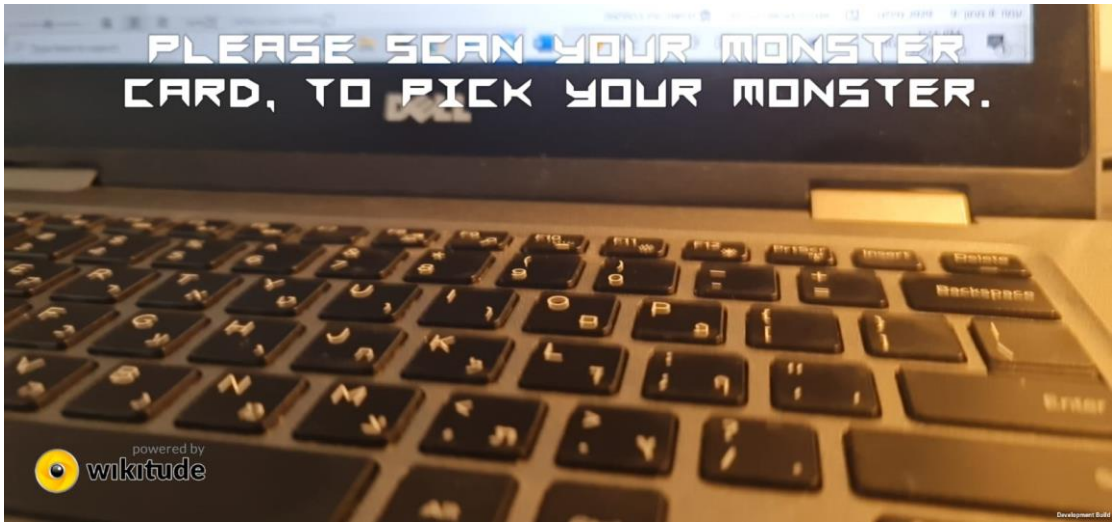

Which in turn created a perfect sync for both players locations over the network. Sadly this method couldn't be applied to the cube how ever since it velocity couldn't exactly be tracked since it is attached to the marker and transfers with it instantly, so the cube synchronization had to be applied in short bursts of teleportation, each time the cube location changes by big enough delta.

Concluding points & ideas for future expansion

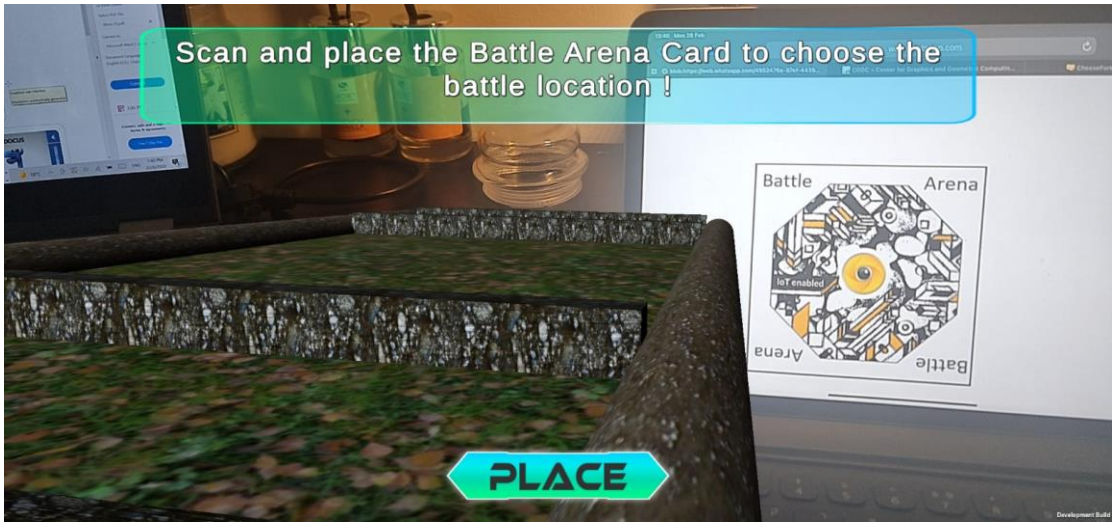
As we have discussed throughout this document, we took a lot of aspects into consideration, plus we were limited by the timetable of the project. But the more that we dived into the project, the more ideas we got for future expansions, that is why we wrote the code in a way that would allow relatively easy additions and expansions:

1. sound effects for interactions: something that can contribute to the game experience is original effects special for our case (we used ready effects from the unity asset store), sound effects can also be a nice addition to have, for different events we could attach different sound effects, in order to achieve more immersive experience.
2. 2v2 battles: we can think of this as an expansion to the original idea, 2 vs 2 games can be even more challenging, and can introduce another changing factor to the game, making it way more dynamic and interesting. (since we would have 4 cubes in total).
3. more arenas and monsters: for now we have one arena, and 4 dinosaurs to choose from, all of the dinosaurs have the same battle statistics, in the future we can have different champions with battle statistics, and tradeoffs between attack speed, movement speed, health, and other attributes, combine this with the 2 vs 2 game, we have a complete game with different strategic approaches.
Most of the backend for this expansion is ready, since the game is built in a way that it can support that, the issue of implementing it instantly would be mainly the balancing issue in order to create a fair game which would need a research by itself.
4. further enhances graphics and visual effects: we were limited both by time and the unity asset store to get our ready-to-use champions. If given the opportunity we could create different special champions and even a better arena.
5. enhance server sync speed and accuracy: instead of using the Photon Network that offers the syncing services, we could also implement the network solution ourselves and achieve better results since we can tailor the solution to our needs exactly, but this could also be a full scope project by itself 😊 .

Actual gameplay:




MONSTER FOUND!
PRESS START TO CONFIRM YOUR SELECTION AND START THE BATTLE!
NO NAME: DIPLODOCUS
HEALTH: 3400
ATTACK DAMAGE: 50
ATTACK SPEED: 2
START


Scan and place the Battle Arena Card to choose the battle location !
Battle
Arena
IoT enabled
PLACE


Arena placed, please Scan your cube, then press start to BATTLE!
Battle
Arena
IoT enabled
ADJUST


Joined to Room587. Waiting for other players...
YOU
3600/3600


YOU
3600/3600


RESPAWNING AT:
4.00


YOU
3600/3600