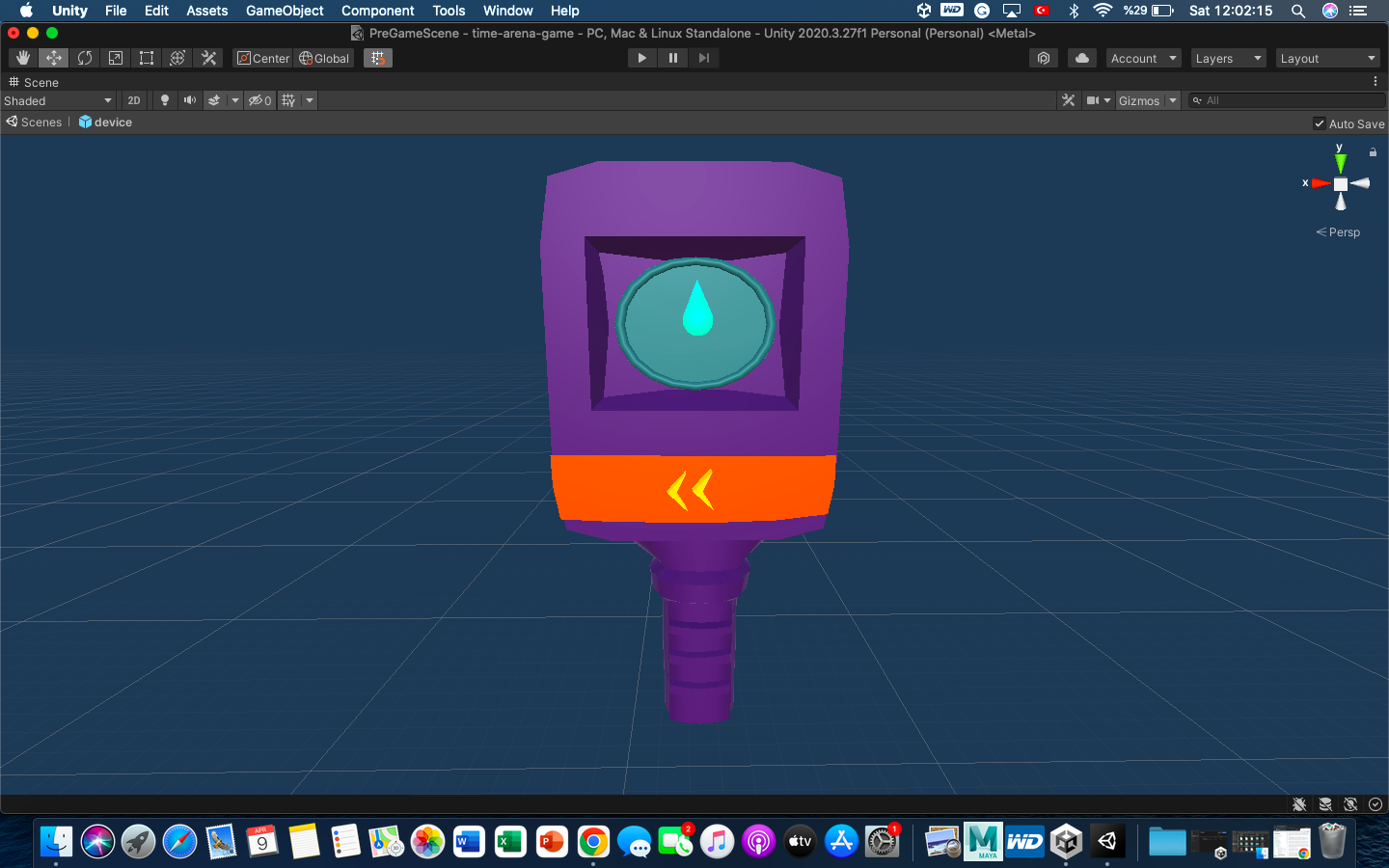
The tracker device has two parts: a compass and a light. The compass has a needle that points towards an objective. The light indicates whether you should time travel and in which direction.



# Version 1 (simple)

The tracker device’s script is called CrystalDevice.cs. Edit its Update() function to do the following.

## Compass Needle

For miners, their objective is the nearest crystal. Only consider the crystal’s position in space, not time. Just use Euclidean distance. The algorithm for this should then be straightforward:

1. Iterate over every crystal currently in the game. The CrystalManager script should have a list of these crystals.
2. Find the one that is closest to the player. You can find the distance between two Vector3s using Unity’s [Distance()](https://docs.unity3d.com/2020.3/Documentation/ScriptReference/Vector3.Distance.html) function.
3. Find the direction from your player’s position to the crystal. This can be done using crystalPosition – yourPosition.
4. Find the direction you are facing using your rotation value.
5. Now find the angle to rotate the needle by:
6. First, convert these vectors into Vector2 values (since we’re only interested in horizontal distance). You can use [this](https://docs.unity3d.com/ScriptReference/Vector2-operator_Vector3.html) method from Unity.
7. Next, use Unity’s [SignedAngle()](https://docs.unity3d.com/ScriptReference/Vector2.SignedAngle.html) method to find the angle between them.
8. Pass the result into our ChangePointerPosition() function. You may have to multiply by -1 if the sign is the wrong way round (i.e. you found an anticlockwise angle instead of clockwise).

For guardians, their objective is the nearest miner. Use exactly the same approach as with crystals, but this time iterate over miner positions. GameController has a dictionary of all miners in the game.

## Light Colour

Once you’ve identified the nearest objective, do the following:

1. Check if the distance (found in step 2 above) is below some threshold. Set this threshold in the Constants script. Here the idea is to check if you’re getting close to your target, i.e. you’re in the same room kind of deal.
2. If it is below the threshold, check where the target object is in time. TimeLord will be able to provide this information.
3. Set the light to OFF/ORANGE/BLUE according to when your perceived frame is relative to your objective. You can do this with the SetButtonMaterial() function.

The way to check these frames will be different depending on whether your objective is a crystal or miner.

Each crystal has a range of frames it exists in as a (float startFrame, float endFrame) tuple. So you’ll need to compare if your perceived frame is before the start frame (set light to BLUE), after the end frame (set light to ORANGE) or in between (set light to OFF).

Each miner will have their own perceived frame, just like yours. So you’ll need to compare if your perceived frame is before theirs (set light to BLUE), after theirs (set light to ORANGE) or equal (set light to OFF).

# Version 2 (more complex)

If I have more time I’ll flesh this out. Adapt the above with the following changes:

## Compass Needle

1. Create a graph representation of the map. I’m sure C# has a graph data structure you can use or feel free to implement one from scratch if you’re feeling ballsy.
2. Create some kind of interface for this graph that allows you to take in a player or crystal’s position and determine where they are in your graph representation.
3. Use a path finding algorithm like Dijkstra or A\* to find the shortest path from your position to an objective in the graph.

## Light Colour

1. For miners, add another check to see if you are near a guardian.
   1. If a guardian is within a certain threshold, set the light to encourage them to time travel.
   2. Access your perceived frame and compare it to current elapsed game time.
   3. If your perceived frame is closer to elapsed time than frame 0, set the light to ORANGE (encouraging them to travel back where there’s more time to hide in).
   4. Otherwise set the light to BLUE.
2. Add another check for both miners and guardians when they are near a dynamic crystal obstacle. Not sure how we are planning on implementing these at the time of writing but I assume we’ll have access to information about the state of the obstacle at a given frame (blocker or non-blocker).
   1. Check to see if your position is within a certain threshold from a dynamic obstacle. If so, do the following.
   2. If the obstacle is not currently blocking your path, set the light to OFF.
   3. If it is, compare your perceived frame to the (hopefully provided) data w.r.t. when the crystal is not a blocker.
   4. Set the light to ORANGE or BLUE accordingly.