## [crystal object] CrystalBehaviour.cs

Store a crystal ID and existence interval as a (float startFrame, float endFrame) tuple.

Every update, check with TimeLord to see if your player’s perceived frame is within the existence interval. If so, set the crystal object’s layer to Player. If not, set the layer to OutsideReality.

## [crystal manager object] Crystal.cs

A tiny helper class (could make it a private inner class with public attributes).

Stores the physical location as a Vector3.

Stores the existence interval as a (float startFrame, float endFrame) tuple.

## [crystal manager object] CrystalManager.cs

Stores a map of crystal IDs to Crystals:

Dictionary<int, Crystal>

Stores a set of spawn locations for crystals:

List<Vector3>

Every update:

* Consider generating a new crystal
  + Only the MasterClient should be allowed to do this
  + Check the number of items in your crystal map (only add a new crystal if the size is below some threshold)
* Check with TimeLord for the current elapsed time frame in the game
* Choose a new existence interval whose start frame is randomly chosen to be between 0 and the current frame, skewed towards the current frame
* Randomly choose a position from the spawn locations
* Instantiate a new crystal object at the chosen position across Photon
* Set its time interval in the CrystalBehaviour script
* Make an RPC call with the position and existence interval on all other clients to generate a new crystal

RPC generate new crystal:

* Add a new Crystal with the given values to the crystal map

## [player object] CollectingCrystals.cs

Only allow collecting crystals if the crystal’s layer is Player.

On collection:

* Tell PlayerController to increment their score
* Tell GameController to increment the total score
* Destroy the crystal across Photon
* Tell CrystalManager to remove the crystal from its map

## [player object] PlayerController.cs

Keep track of your score.

Have a jump timer initialised to -1.

On grabbing a player (if you’re a guardian):

* Reset your cooldowns to 15f

On being grabbed (if you’re a miner):

* Tell PlayerController to decrease total score by half your score
* Reduce your score by 50%
* Randomly choose a direction (Forward or Back)
  + Maybe choose whichever one has more time for you to explore? (access this info via TimeLord)
* Call your TimeJump() function with this direction and out=true
* If Forward was chosen, ask TimeLord for the time between your perceived frame and current game time
* If Back was chosen, ask TimeLord for the time between the start of the game and your perceived frame
* Set your jump timer to a random value within this time (skewed towards 0)

Every update:

* If jump timer > -1 then reduce it using time delta or whatever it’s called
* If jump timer = 0 then call your TimeJump() function with your \_jumpDirection and out=false