Celine de Wijs
3013898

Treasure Hoard

DOCUMENTATION

What is Treasure Hoard?

Treasure Hoard is a multiplayer game in which you play as a sail ship. You are sailing the seas looking for treasure chests, but you are not the only one. Other players want the same treasure chests as you, to earn as much points as possible. They will try to shoot you and to prevent yourself from dying, you have to kill them before they kill you.

How it works

You can navigate the seas using your WASD or arrow keys. You can shoot other ships by using your left mouse button.

When you die, and you have logged in on http://studenthome.hku.nl/~celine.dewijs/kernmodule/login.php, then your score is send to the database. On http://studenthome.hku.nl/~celine.dewijs/kernmodule/highscore_choice.php.

Process

My initial idea was to make a turn-based battle game. I had been working on this game for weeks, but I kept being stuck on the same problem: the turns wouldn't synchronize correctly between server and clients. I've tried fixing it for a very long time, with help from classmates, but to no avail. At one moment (pretty close to the deadline) I decided to make a new game. This, unfortunately, didn't turn out to be a turn based game, but this process has taught me a lot about Unity Networking.

For this entire process I have used tutorials by Brackeys and Ben Tristem on Udemy.

Update: I've made the game turn-based and updated the UI system. Collecting the treasure chests is also much safer now, because it's not done on the client.

UML Diagram

PlayerSetup

- componentsToDisable: Behaviour[]
- remoteLayerName: string

- Start(): void
- AssignRemoteLayer(): void
+ OnStartClient(): void
- DisableNonLocalComponents(): void
- OnDisable(): void

GameManager

+ instance: GameManager
+ players: Dictionary<string, Player>
- PLAYER_ID_PREFIX: const string

- Awake(): void
+ RegisterPlayer(string, Player): void
+ UnregisterPlayer(string): void
+ GetPlayer(string): Player

PlayerShoot

+ damage: int
+ range: float
- mask: LayerMask

- Update(): void
- Shoot(): void
- CmdPlayerShot(string, int): void

PlayerUI

- player: Player
- canvas: Canvas
- score: Text
- slider: Slider

- Start(): void
+ UpdateScore(): void
+ UpdateHealth(): void

Player

- disableOnDeath: Behaviour[]
- startHealth: float
- currentHealth: float
- isDead: bool
- score: int
- playerUI: PlayerUI

- Start(): void
+ ResetPlayer(): void
+ GetCurrentHealth(): float
+ GetScore(): float
+ AddScore(int): void
+ RpcTakeDamage(int): void
- Die(): void

ScoreManager

- secretKey: string
+ addScoreURL: string
+ highscoreURL: string
+ game: string

+ PostScores(int score): IEnumerator
+ Md5Sum(string): string

Spawner

+ prefab: GameObject
+ pooledAmount: int
+ minX, maxX, minZ, maxZ: float
+ offset: float
- objectPool: List<GameObject>

+ OnStartServer(): void
+ Spawn(): void
+ SetInActive(GameObject): void

ObjectInteraction

- player: Player
- spawner: Spawner
- scoreAmount: int

- Start(): void
- OnTriggerEnter(Collider): void
- CmdPickup(GameObject): void
- RpcPickup(GameObject): void

PlayerMovement

- inputTransform: Vector3
- inputRotate: Vector3
- sailSpeed: float
- rotationSpeed: float

- Update(): void