Under Amsterdam

Technical manual

January 2023

Adrien Ploix-Noguer
Daniël Vermeulen
Jeroen Officier
Luan Praud
Quinn Koene

ISSUED BY
VRAholics

Tile System

- CubeInteraction

  Core of the grid functions, manages the pipes enabling, previewing, connecting and so on.

  - GetNeighbors()

    Fetches the tiles' neighbors' game objects and scripts in all 6 directions (up, down, right, left, forth and back).

  - UpdateNeighborData(bool enable, string playerCompany = "")

    Checks if the neighbors' and player's companies match to see if they are eligible to connect with the tile the player is willing to place.

  - VerifyRules(string pCompany)

    Checks if the tile the player is willing to place respects some safety proximity rules with others companies' nature.

  - EnableTile()

    Enables the tile placed by the player by rendering it and changing its company to the players.

  - DisableTile()

    Breaks the tile when hit by a hammer by disabling it and resetting its state and company to "Empty".

  - OnRenderPipePart(bool isActive)

Enables the pipe's visual according to its matching neighbors and company.

➢ OnRenderPipePreview(bool isActive)

Enables the pipe's preview according to its matching neighbors.

➢ CheckConnectionForWin()

Function that goes through tiles of the same company to verify if they are describing a continuous path between output and active inputs.

❖ ioScript

Manages the input and output tiles (IOTiles) enabling.

➢ StartCheckingPipes()

Function that launches the verification procedure of connected output to active inputs.

➢ AddPlayerOutputs()

Adds Output tiles for each company at the beginning of the game.

➢ AddPlayerInputs()

Adds Input tiles for each player at the beginning of each round.

➢ PlaceIOPipe(string company, bool isOutput)

Place input and output pipes in a random location on the walls.

❖ IOTilesScript

Regroups functions and data of output and input tiles.

➢ GetNeighbours()

Fetches the potential IOTiles neighbors' of the tile in each 6 directions.

➢ TryEnableIOPipe(string setCompany, bool shouldBeOutput, bool isSyncing)

Verifies if the IOTile can be enabled here, renders it and instantiates indicators.

➢ SpawnIndicator(bool shouldBeOutput)

Instantiates the indicator arrow and the smoke particle effect on the corresponding IOTiles for the player.

❖ PipeColouring

Update the color of the placed pipes depending on the player company.

➢ UpdateRenderer(string pipeCompany, GameObject givenGO = null)

Get a call when a tile is enabled and change the color of each part of the pipe corresponding to the company.

Handles fusion XR host

- Room name (Room configuration)
- Fusion settings

- Local user spawner
- Event
- Maximum players
- Input authority
- Keeping track of player ID
- User despawn after disconnect

Gamemanager

Manages the order of execution of the game, unity events

- Takes care of amount of rounds
- Scene switching
- Points manager
- On game start
- Randomactivatedroots
- On countdown start
- On countdown end
- On round start
- On round end
- On round late end
- On game end

❖ CompanyManager

Gives company to players and keeps track of it.

➢ GetCompany(PlayerRef player)

Gets a free company to the player referring to his history.

➢ loadSend()

Sends company to each player in the game and stores the information in a dictionary.

- ResetCompanies()

  Resets the company's dictionary at the end of each round.

❖ Pointsmanager

Changes the amount of points of a player determined by his actions.

- AddPoints(string company)

  Adds points to the player after removing a tile.

- TeamworkBonus(string company)

  Adds Bonus points related to teamwork.

- RemovePoints(string company)

  Removes points for the cost when placing pipes.

- RemovePointsRoots(string company)

  Removes points for the cost of removal of the roots.

- CalculateRoundPoints(string company)

  Rewards the players with a non negligible amount of points when succeeding in connecting Output to active input.

❖ TeamworkManager

Manage TeamWork contracts between players if they handshake.

- AddTeamWork(string company1, string company2)

Create a contract between two players if they aren't in a contract already. Is called by TeamworkHand script on collider trigger.

➢ CheckMyCompany(string company)

Check if I am in a contract with another player, if so, return the other player

➢ CompanyDone(string company)

Is called from CubeInteraction(CheckConnectionForWin), if a company is done, save their company. Save the player associated with the company to a Dictionary as Key or Value, if we don't already exist as one.

➢ CheckTeamwork()

Is called on RoundLateEnd from Gamemanager. Goes through all players who are done and checks if their teammate is also done. If so, grant both of them bonus points.

➢ CheckFree(string company)

Check if the player is free to work with someone else.

❖ ColourSystem

Applies company materials to gloves, watch, hat and placed pipes (Mostly called by PipeColouring.cs)

➢ SetColour(GameObject givenGO, string companyName)

❖ CityMove

Makes the player move down the game area when the game starts

➢ CheckAllPlayers()

Checks if every player connected is ready (by grabbing a hat).

➢ MovePlayers(Vector3 from, Vector3 to)

Coroutine that plays the animation moving players from lobby to digsite at the start of the game and from digsite to lobby at the end of the game.

➢ DisableObjectsAfterGameStart()

Disables some objects of the lobby scene after the game starts to alleviate the renderer

➢ EnableObjectsAfterGameOver()

Enables back the disabled objects of the lobby scene.

❖ Environment script

Adds unmovable rocky obstacles to the game

➢ RPCSpawnRock()

Instantiates a random amount of rocks between 1 and 3 at a random location on the map.

➢ RPCDestroyRock()

Destroys all the rocks at the end of the game.

❖ Scoreboard

Ranks the players on a board according to their points.

➢ UpdateLeaderboard()

Fetches the players' points and stores them in a dictionary to sort them by descending.

➢ DisplayLeaderboard()

Displays all scores on the leader board game object at the end of the game.

➢ WarpPlayers()

Warp players on the podium according to their ranking in the leaderboard at the end of the game.

❖ VisualCountDownLocal

Manage the start countdown and the game Over text.

➢ StartLocalCountDown()

Display 3,2,1, Start!, before the first round start.

➢ ToggleDisplayGameOverText()

Enable the game over message and call the “DisableGameOver” coroutine.

➢ IEnumerator DisableGameOver()

Coroutine that disables the text after a certain amount of time.

❖ Connection Manager

This is Fusion's base connection manager that handles the async Connect() function. Make sure you add a component to the runners callbacks with runner.AddCallbacks(this) if you are trying to run INetworkRunnerCallbacks.

➢ OnSceneLoadDone()

This function loads when a network scene is done loading. We use it to set some references and set the player position and un-fade the players vision.

➢ OnPlayerJoined()

Fusion function that adds players to active player list and makes a network dummy that is visible for other players.

➢ OnPlayerLeft()

Is called whenever a player leaves a connection. We remove them from the list of active users.

- OnShutdown()

  If the client's network manager disconnects it destroys itself and calls shutdown, we use it to move the player back to the menu scene.

- OnConnectedToServer()

  If client is connected to server load new scene

Player and Inputs

- PlayerData

  Stores points and company of the player. Has useful functions

  - UpdatePlayer(Changed<PlayerData> changed)

    Static function that changes the company materials when this one is updated.

  - RPC_SwitchHands()

    Switches the hands preferences and moves the watch to the other hand.

- WristMenu

  Displays information on the watch such as company, timer, amount of points of the player.

  - ChangeImage

    Updates company image on the watch.

  - winLosePoints

Instantiates the particles' effects when winning or losing points.

❖ SettingsUI

Settings user interface to set the game volume or the hands preferences.

➢ SetVolume(string sliderType, float sliderValue)
➢ MasterVolume()
➢ LeftHanded()

❖ HandTileInteraction

Manages the hands inputs and the interactions with the tiles

➢ FixedUpdateNetwork()

Get the players inputs for grabbing hammer or placing tiles.

➢ OnTriggerEnter(Collider other)

When touching a tile, enables its preview if it’s not occupied.

➢ OnTriggerExit(Collider other)

When exiting a tile, disables its preview if it’s not occupied

➢ SwitchHands()

Switch hands preference (One enables the tiles, the other has the watch and grabs the hammer)

❖ HammerScript

Player’s tool that can break tiles

➢ RPC_DisableRoot(NetworkObject touchedRoot)

Breaks the hit root by disabling it.

➢ OnTriggerEnter(Collider other)

Verifies if the player and tile companies match and if the hammer has enough velocity to break.

❖ PlayerTestMove

Move the player depending on the joystick movement.

➢ SwitchMovementControls

Switch which controller (left or right) can move the player.

❖ TeamworkHand

Verify if 2 players touch their hands and add a teamwork contract to the TeamworkManager, and activate particles.

➢ OnTriggerEnter(Collider other)

When the hands of both players collide above the head of the player, they have a high five and call RPC_SendParticle, and add a contract to the TeamworkManager script.

➢ RPC_SendParticle()

Call a particle effect on the handshake location for each client.

Tree Roots

Tree roots are procedurally generated meshes along Bézier Curves, the following parameters are fully customisable : the base radius, the end radius, and the number of vertices in the Unity inspector.

- BezierPoint

  Struct that stores the position and rotation of a Bézier point

  - BezierPoint (Vector3 pos, Vector3 forward)

    Constructor that sets the position and the rotation thanks to a forward vector.

  - LocalToWorld(Vector3 localSpacePos)

    Returns the world position of the point.

  - LocalToWorldVec(Vector3 localSpacePos)

    Returns the world vector.

- Circle2D

  A scriptable object which creates a local circle 2D mesh vertices list, you can change the number of vertices with the “angularCounts” parameter.

  - UpdateVertices(float t)

    Fill the vertices and an uv list of the 2D mesh depending on t.

❖ RootSegment

Script that generates the complete mesh of the root

➢ GenerateMesh()

Generates all the 3D mesh of a root by filling the mesh with the vertices, UVs, normals and triangle lists.

➢ GetBezierPoint(float t)

Function that returns a BézierPoint depending on the 4 control points of the curve at a “t” time.

❖ RootGrower

➢ IEnumerator Grower(Material mat, rootS root)

Coroutine that makes the roots grow by controlling the “GrowStep” parameter of the RootShader.

❖ RandManager

➢ randomActivatedRoots ()

Executed only by the host, called at the game start, generates a random index to choose which root will be enabled. Call RPC_RandRoot().

➢ RPC_RandRoot ()

Executed by every clients, and activates the roots GameObjects from the list thanks to the index from “randomActivatedRoots”

Valve

The Valves scripts control the 3 difficulties valves and the ExitValve.

- ❖ Valve
  - ➢ valveTurned()

    If the valve is turned more than 90°, invokes the valve turned event.

- ❖ ExitValve
  - ➢ ReturnToMenu()

    Returns to the Menu scene if the valve turned event is invoked.

  - ➢ OnTriggerExit(Collider other)

    Despawns the Exit Valve if you are too far from the Valve.

- ❖ SetDifficulties
  - ➢ MoveSelection()

    Make the selection indicator object move and rotate to his position when the difficulty valve is selected.

  - ➢ RPCValve()

Executed by every player when the “ValveTurned” event is invoked. Set the game settings corresponding to the valve difficulty.

➢ RPCChangeValve()

Reset and deactivate all the valve selection objects when the current selected valves change.

❖ ExitValveSpawner

➢ SpawnPipe()

Instantiate the ExitValve prefab in front of the player.

➢ MovePipe(GameObject pipe, Vector3 targetPos, Vector3 pos)

Move smoothly the pipe outside the ground

➢ DespawnPipe(GameObject pipe)

Despawn the valve given in parameters.