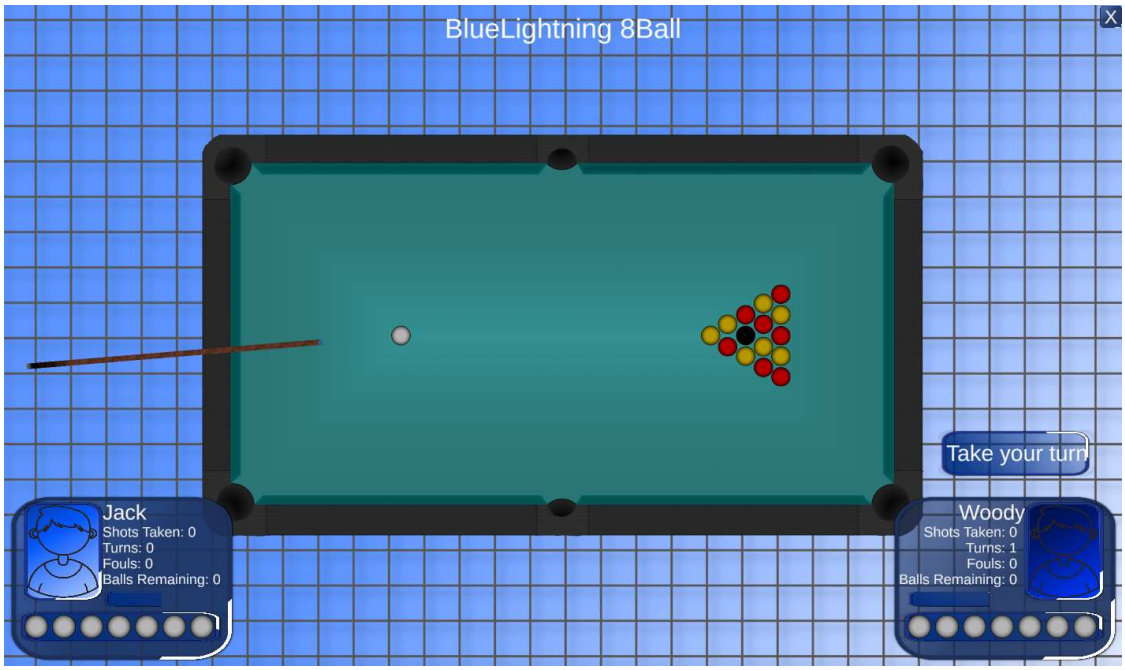
About

The TegridyPool package provides a basic 2D pool framework for deploying your own Pool/Snooker inspired games. Levels, balls and players can be configured in a number of ways enabling a great deal of variety from this simple controller.

Usage

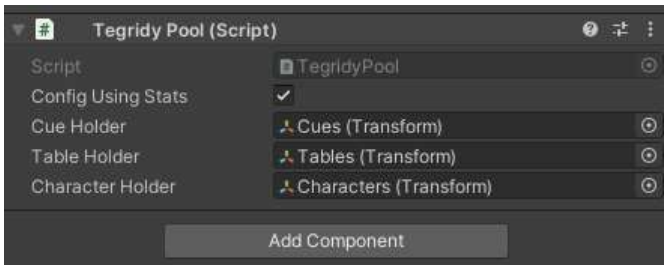
We recommend taking a look at the example scene to get an idea of how the scene is composed and setup. You will have two main objects, one a GUI Canvas for the menu's and overlays and the other will contain your tables, cues and characters. This last object should be left active when you call StartUp on the TegridyPool component. If you have the StandAlone component configured in your scene the menu will load automatically, if not you will need to call the TegridyPool component in your script as shown in the StandAlone script. The package also contains blueprints for a complete game ready to be started, and the main components for making your own levels. I would recommend looking at these blueprints before creating your own.



Support:

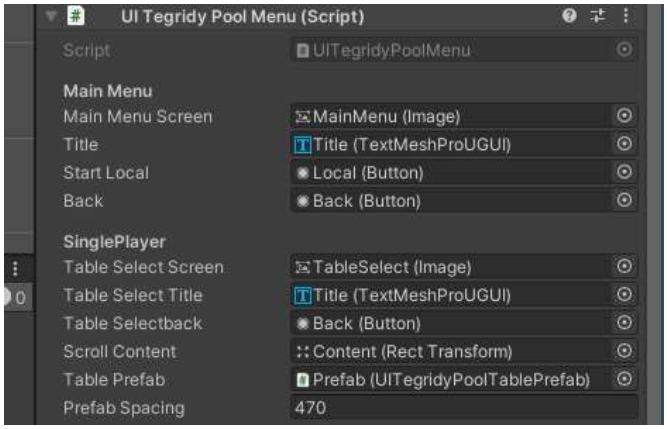
support@tegridygames.co.uk

TegridyPool Configuration



- **ConfigUsingStats** – If this is set the players strength & accuracy will be based off the stats sent for the character, if this is false all characters will have maximum stats.
- **CueHolder** – This should be set to the parent object that contains all the *TegridyPoolCue*'s in the scene.
- **TableHolder** – This should be set to the parent object that contains all the *TegridyPoolTable*'s you have configured in the scene.
- **CharacterHolder** - This should be set to the parent that contains all the *TegridyPoolCharacter*'s that have been configured in the scene.

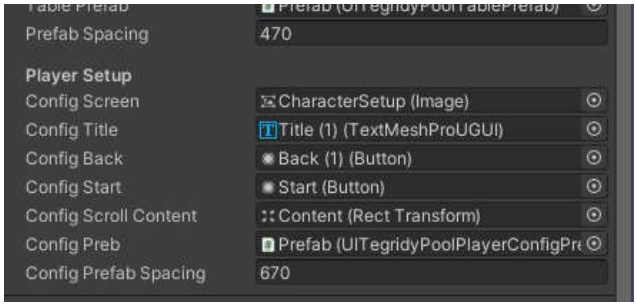
Menu Configuration



- **MainMenuScreen** – This should be set to an image component that contains the objects used on the main menu.
- **Title** – This should be set to a TextMeshPro object set as a child to the *MainMenuScreen* and will display the games name if used.
- **StartLocal** - This should be set to a Button object set as a child to the *MainMenuScreen* and will be used to move onto the level select screen.
- **Back** – This should be set to a Button object that is set as a child to the *MainMenuScreen* and will either quit the aplication or enable the host menu is the value was set when OpenMainMenu() from TegridyPool.
- **TableSelectScreen** – This should be set to an image that is the parent of the GUI objects used in the *TableSelectScreen*
- **TableSelectTitle** – When found this is used to display the table select heading found in the language controller.
- **TableSelectBack** – This should be set a Button object and will take the player back to the main menu. If a TextMeshPro object is found it will also use the

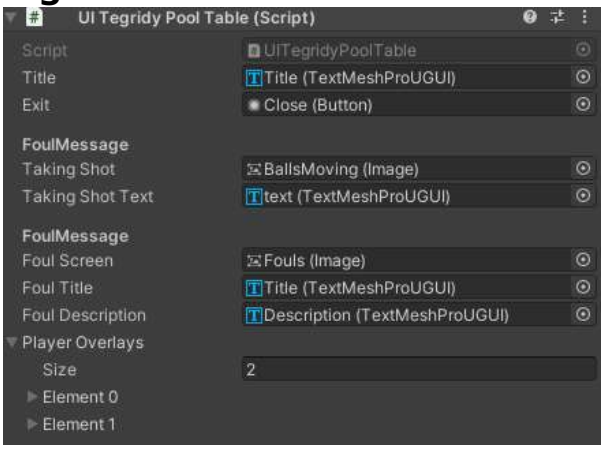
relevant strings found in the language controller.

- **ScrollContent** – This should be set to the content of a scroll view component and contain a prefab to be used in the level select
- **TablePrefab** – This should be set to an image containing the GUI object that make up the level information
- **PrefabSpacing** – how far apart the prefabs should be spaced apart when generated by the application.



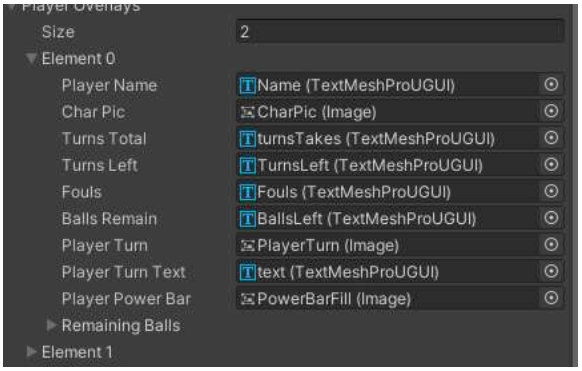
- **ConfigScreen** – Should be set to a image object that contains all the components that make up the character and cue select screens.
- **ConfigTitle** – Used to display the screen heading if a TextMeshPro object has been defined.
- **ConfigBack** – Should be set to a Button object and navigates back to the level select screen. If a TextMeshPro object is found as a child, this will be set using the TegridyPoolLanguage Class
- **ConfigStart** – Should be set to a Button object and will start the round. If a TextMeshPro object is found as a child, this will be set using the *TegridyPoolLanguage* Class.
- **ConfigScrollContent** – This should be set to the content of a ScrollView Component and will be used for holding the character selection prefabs.
- **ConfigPrefab** – These will be drawn for each of the players set in the tables config and are used for setting the players character and cue.
- **ConfigPrefabSpacing** – The spacing between the spawned prefabs.

Player Overlay Configuration



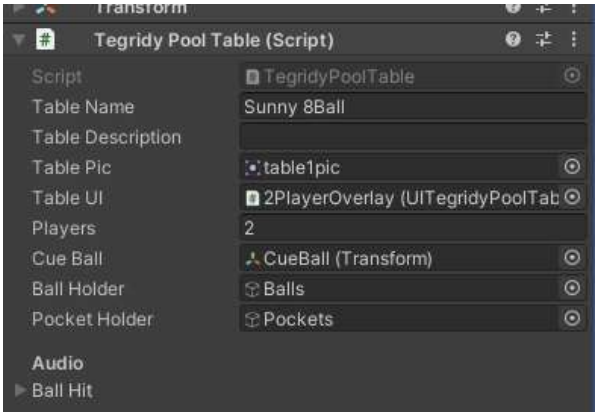
- **Title** – If found it will be used to display the level name.

- **Exit** – Button object used to exit the level if the player wants to quit.
- **FoulMessage** – Set to an image object,used for displaying match status updates
- **FoulTitle** – Should be a child of *FoulMessage* and set to a TextMeshPro object, used for displaying the heading of any messages.
- **FoulDescription** – Will display any important info regarding the last shot taken. Should be a child of *FoulMessage.*



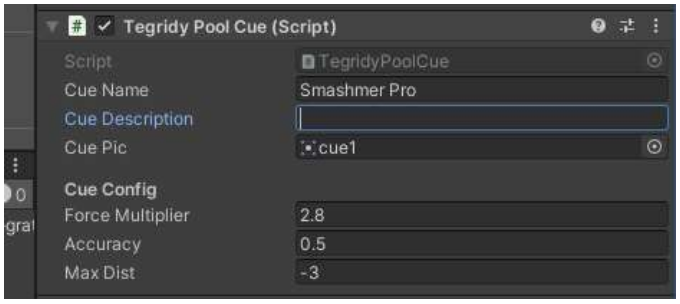
- **PlayerOverlays** – This should be set to the number of players on the table with each element being configured.
- **PlayerName** – If found this will display the chosen characters name in the GUI and should be set to a TextMeshPro Text object.
- **CharPic** – This will be used to display the chosen characters picture.
- **TurnsTotal / TurnsLeft / Fouls / Balls Remain** – used to display info about the current game.
- **PlayerTurn** – Should be set to an image object and is used to notify the player which character is taking there shot.
- **PlayerTurnText** – This string is defined from the language controller and is used to indicate which player is taking there shot.
- **PlayerPowerBar** – This should be set to an image object that has its type set to filled and is used to display the current shots power.
- **RemainingBalls** – This should be the same size or larger than the amount of balls on the table and have its image elements defined. These images will be swapped with the players balls once the have selected a colour.

Table Configuration



- **Table Name / Description** – If the number of languages in the scene matches the length of the arrays in *TegridyPoolLanguage* then it will use the strings defines there instead of what it finds in the editor.
- **TablePic** – This image will be used on the level select screen and should represent your level.
- **TableUI** – This will provide an overlay for the table, the overlay used should have the same amount of players and balls as the table does.
- **Players** – How many players are on the table, your ball ID's should go one higher otherwise a final ball will not be found.
- **CueBall** – A game object with a *TegridyPoolCueBall* component and a 2D collider.
- **BallHolder** – the parent object that contains the balls for this table. All balls should have a *TegridyPoolBall* component added, and the balls group ID should end one higher than the player count. The final ID will be used as the last ball the players will need to pot.
- **Pocket Holder** – the parent object that contains the pockets that the balls will interact with on that table and each of these should have a *TegridyPoolPocket* component.
- **Audio** – The audioclips to be used at different stages in the game.
- For each table you will also need to add a sprite object with a collider that represents the tables boundaries. And one without a collider that will act as the table itself.

Cue Configuration



Each TegridyPoolCue component should be placed on a sprite component that represents the in game pool cue complete with an AudiosSource.

- **CueName / Description** – Can be set in the editor, or if the number of cues in the scene matches the length of the arrays in TegridyPoolLanguage then they will be used instead to provide the GUI's strings.
- **CuePic** – Should be set to a sprite and will be used in the GUI when selecting the characters pool cue.
- **ForceMultiplier** – If the application is using character stats then this is used to multiply the characters Strength.
- **Accuracy** – How likely the player is to hit straight and true.
- **MaxDist** – How far the cue can move back In the game when the player is setting there power.

Character Configuration

- **Character Name/Descritpion** – If the amount of characters in the scene is equal to the length of the arrays found in *TegridyPoolLanguage* then these will be define from there instead of in the editor.
- **CharacterPic** – A sprite representing the players character, this is displayed on the character select screen.
- **Strength/Luck/Dexterity** – Used when calculating the players shot if *configureUsingStats* has been set on the *TegridyPool* component.

Script / Class Descriptions

- **StandAlone** – This is used to start the game when you are only using the TegridyPool component and are not importing it into another project.
- **TegridyPool** – StartUp should be called when your application starts, this will configure the language and discover the game objects in the scene. After StartUp has been called you can open the main menu by passing OpenMainMenu a configured GUI, host can be left null but if it is set that object will be disabled when the menu is opened and set active again once the game has been quit usefull if implementing into your own systems.
- **TegridyPoolBall** – Added to the pool balls to keep track of collisions, provide a ID for the ball and a audio clip for when the ball is hit.
- **TegridyPoolCharacter** – Added to a empty game object and provide stats and character info.
- **TegridyPoolCue** – Script for controlling the players cure when the game is in play.
- **TegridyPoolCueBall** – Script for keeping track of the cue ball interactions.
- **TegridyPoolLanguage** – Provides easy access to all strings used in the GUI
- **TegridyPoolPocket** – Added to each pocket on the table and keeps track of any balls that enter a pocket, also provides a audio clip that is used when the ball is removed.
- **TegridyPoolTable** – Should be added to each table you have in your scene, provides configuration options and settings for the tables audio.
- **TegridyPoolWall** – Should be added to all off your table boundaries, provides audio on collision with a ball.
- **UITegridyPoolMenu** – Should be added to the parent of your main menu, Holds the GUI elements that the game will use for its menus.
- **UITegridyPoolPlayerConfigPrefab** – Holds the GUI elements used for the

character select screen.

- **UITegridyPoolTable** – Provides an in game overlay to keep track of the game, this should contain enough player info elements for the amount of players on your table.
- **UITegridyPoolTablePrefab** – This component holds the GUI elements used for creating the level select option.