


Kit Patcher Pro



v0.8
Ghostdog Studio

- 
- I. **Overview (p2)**
 - A. **Description**
 - II. **Requirements (p2)**
 - A. **Unity Version**
 - B. **Dependencies**
 - 1. **Core.Renci.SshNet**
 - 2. **TextMeshPro (TMP)**
 - III. **Specifications (p4)**
 - A. **FTP Connection Initialization**
 - B. **Folder Downloading**
 - C. **Asynchronous File Download**
 - D. **Version Checking**
 - E. **File Content Comparison**
 - F. **Integration with Existing Functionality**
 - G. **File Update Checking**
 - H. **Local File Metadata Caching**
 - I. **Update Progress Display**
 - J. **Game Launching Integration**
 - K. **Error Handling**
 - L. **Optimization for Performance**
 - IV. **Menu & Buttons (p6)**
 - A. **Home**
 - B. **Events**
 - C. **Community**
 - D. **Account**
 - E. **Support**
 - F. **Patch Notes**
 - G. **Update**
 - H. **Launch**
 - V. **Documentation (p11)**
 - A. **Game Launcher Prefab**
 - B. **Sub-Folder & Folder Name**
 - C. **KitPatcher**
 - D. **ServerStatus**
 - E. **UIMenuController**
 - F. **UIButtonSound**
 - G. **Kit Integration**
 - VI. **F.A.Q. (p18)**
 - A. **Game Launch Issues**
 - B. **Download Interruption**
 - C. **Game Directory Movement**
 - D. **Integration into Unity Project**
 - VII. **Version Log (p19)**
 - A. **v0.2**
 - B. **V0.3**
 - C. **V0.4**
 - D. **V0.5**

Overview

The "**Kit Patcher/Launcher**" is a lightweight and user-friendly game launcher which offers essential features such as real-time server status checking, game file installation and updates, patch notes, menus, and seamless game launching. Players can easily stay informed about server news and availability, download the latest updates, and access patch notes and other menus before launching the game.

Requirements

Core.Renci.SshNet

CompactStandaloneFileBrowser

TextMeshPro (TMP)

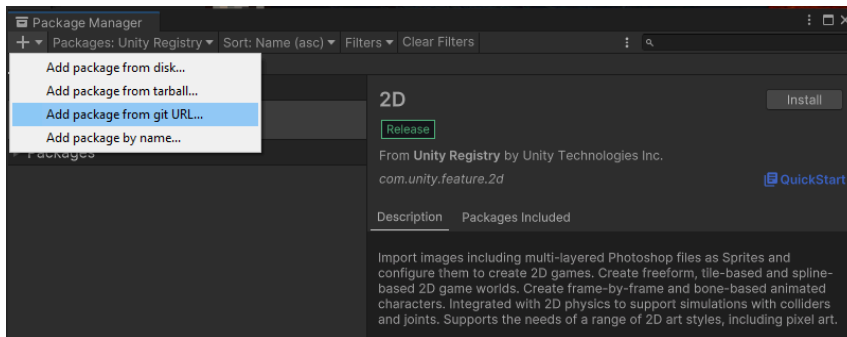
Renci.Ssh.Net

1. **Download :** <https://www.nuget.org/packages/Core.Renci.SshNet/>
2. **Extract the Contents:** First, extract the contents of the .nupkg file. You can do this by changing the extension of the file from .nupkg to .zip and then extracting it using a tool like WinZip, WinRAR, or any other ZIP file extractor.
3. **Locate the DLL:** After extracting the contents, you should find the Renci.sshnet DLL for the Renci.SshNet library. This file is typically located within the **lib** folder. You want to choose the **Net40** DLL.
4. **Copy to Unity Project:** Copy the necessary Renci.SshNet DLL into your Unity project's Assets folder. Make sure to organize into appropriate folders within the Assets directory if necessary.

Compact Standalone File Browser

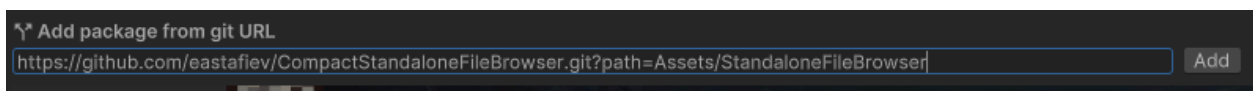
INSTALL URL:

1. **Navigate to Package Manager:** Go to **Window > Package Manager** to open the Package Manager window.
2. **Add a Git Dependency:** In the Package Manager window, click on the **+** button located in the upper-left corner. Then select **Add package from git URL....**



3. **Enter Git URL:** Enter the URL of the Git repository containing the Unity package. Ensure that it's a valid Git URL pointing to the package you want to install.

<https://github.com/eastafiev/CompactStandaloneFileBrowser.git?path=Assets/StandaloneFileBrowser>



4. **Install the Package:** After entering the URL, press **Add** to add the package. Unity will then download and install the package from the provided Git URL.

If you are unable to perform the git url install, you can download the files from Github and drop into your asset folder.

TextMeshPro (TMP)

1. **Access Package Manager:** Once your project is open, go to the "Window" menu at the top of the Unity Editor, then select "Package Manager". This will open the Package Manager window.
2. **Find TextMesh Pro Package:** In the Package Manager window, locate the search bar at the top. Type "TextMesh Pro" into the search bar. The TextMesh Pro package should appear in the list below.
3. **Install TextMesh Pro:** Click on the TextMesh Pro package in the list. You should see an "Install" button. Click on it to start the installation process.
4. **Import TMP Essentials:** After installing TextMesh Pro, you need to import the TMP Essentials. To do this, go to "Window" > "TextMeshPro" > "Import TMP Essential Resources". This will import essential TMP resources into your project.
5. **Setup TMP Font Asset:** To use TextMesh Pro, you also need to create or import a TMP Font Asset. You can import fonts by going to "Window" > "TextMeshPro" > "Font Asset Creator". Follow the instructions to import fonts and create Font Assets.

Specifications

The **KitPatcher** script facilitates the downloading/updating of game files from a remote FTP server to a local directory on the user's PC, ensuring players have the latest game version. Here's a comprehensive technical breakdown of its functionalities:

FTP Connection Initialization:

- Establishes a connection to the FTP server using the provided server address, username, and password.
- Utilizes the Renci.SshNet library for SFTP (SSH File Transfer Protocol) communication.

Folder Downloading:

- Downloads an entire folder and its contents from the FTP server to users local directory
- Supports recursive traversal to retrieve files from subdirectories within the target folder.

Asynchronous File Download:

- Implements asynchronous file download operations to enable concurrent downloads and maximize throughput.
- Utilizes C#'s asynchronous programming features (async/await) for non-blocking I/O operations.

Version Checking: The comparison between the remote and local version files ensures that the script only proceeds with downloading files if there's a difference in the version numbers. This helps prevent unnecessary file downloads when the versions match, indicating that the files are already up to date.

Integration with Existing Functionality: The version checking logic is integrated into the existing download process. It serves as the initial step before initiating the folder download coroutine, ensuring that the version check is performed before attempting to download any files.

File Update Checking:

- The checksum method of file checking involves generating a unique identifier, known as a checksum, for each file on both the local system and the remote server. This checksum is calculated using the MD5 cryptographic algorithm.
- Determines whether each file needs to be updated based on the comparison results.

- Skips downloading files that are up to date, minimizing unnecessary data transfer.

Local File Metadata Caching:

- Maintains a dictionary to cache the last checksum of local files.
- Checksums are stored alongside the file paths for efficient comparison during update checks.

Update Progress Display:

- Updates a TextMeshPro text UI element to provide real-time progress feedback to the user.
- Displays the name of the currently downloading file, enhancing user awareness and transparency.

Game Launching Integration:

- Offers a method to launch the game executable directly from the launcher.
- Initiates the game executable after the file update process is successfully completed.

Error Handling:


- Implements robust error handling mechanisms to manage exceptions during FTP operations.
- Logs detailed error messages to the Unity console for diagnostic purposes and user feedback.

Optimization for Performance:

- Utilizes asynchronous I/O operations and parallel processing to enhance download speed and responsiveness.
- Employs caching strategies to minimize redundant file checks and improve overall efficiency.

Menu & Buttons

The menu and buttons are controlled by the **MenuManager**. These buttons can be easily programmed with sub-menus for developers to add custom content from their game. The Demo scene comes set up with functional buttons. It's recommended to copy and rename the Demo scene if you plan on using it, as future updates may overwrite the scene. If you're



building the launcher from scratch using the prefabs, It's recommended to make copies of the prefabs instead of using the originals.

Home

The home menu serves as the central hub for the launcher. It provides essential information and quick access to various features and sections.

Events

The events menu could display ongoing and upcoming in-game events, such as special challenges, tournaments, or limited-time promotions.

Community

The community menu facilitates interaction and engagement among players outside of the actual gameplay. It often includes forums, social media links, and community-driven content.

Account

The account menu provides access to player account settings, profiles, and management features.

Support

The support menu offers a space for assistance, guidance, and troubleshooting resources for players encountering issues or seeking help.

Patch Notes

The Patch Notes window in the Kit Patcher/Launcher system is an interface element for players to access important updates and changes.

Update

Check/Update/Download current game files from server

Launch

Launches game executable

Install

Install the game files

Classes

KitPatcher Class

Methods

Start():

- Sets the screen resolution, starts coroutines to update version text and check for updates, and sets the current version.

GameInstalled()

- Checks whether the game is installed by verifying the existence of both the game executable file and the version file.
- If both files are found in the specified directory, the method returns **true**, indicating that the game is installed; otherwise, it returns **false**.

UpdateInstallButtonState()

- Manages the visibility state of the install button based on whether the game is installed or not.
- It first checks if the game is installed by calling the **GameInstalled()** method. Then, it sets the visibility of the install button (**installButton**) based on the inverse of the **gameInstalled** flag.
- If the game is not installed (**!gameInstalled**), meaning the flag is **true**, the install button is activated (**SetActive(true)**); otherwise, if the game is installed (**gameInstalled** is **false**), the install button is deactivated (**SetActive(false)**), indicating that there's no need to install the game.

CheckForUpdatesCoroutine():

- This coroutine checks for updates by comparing the local and remote version files.
- It downloads the remote version file and compares its content with the local version file.
- If an update is available, it updates the UI to indicate that.

SetVersionText():

- Sets the version text in the UI by appending the current version to the text.

SetCurrentVersion():

- Sets the current version by reading the version.txt file.
- It constructs the path to the local version file, reads its content, and updates the current version.

UpdateVersionTextCoroutine():

- This coroutine updates the version text in the UI after a short delay.
- It ensures that other operations in the Start method complete before updating the version text.

DownloadFolder():

- This method initiates the download process for the game files.
- It sets up the local folder path, clears the local file checksums cache, and starts the CheckVersionAndUpdate coroutine.

CheckVersionAndUpdate():

- This coroutine checks if an update is required and updates the game files accordingly.
- It compares the local and remote version files, downloads the updated files if necessary, and updates the local version file.

DownloadFolderCoroutine():

- This coroutine downloads the game files from the FTP server.
- It creates the local directory if it doesn't exist, downloads each file asynchronously, and updates the progress text.

GetFileChecksum():

- Calculates the checksum (MD5 hash) of a file on the FTP server.
- It downloads the file, computes the checksum, and returns it as a string.

GetLocalFileChecksum():

- Calculates the checksum (MD5 hash) of a local file.
- It reads the file, computes the checksum, and returns it as a string.

LaunchGame():

- This method is responsible for launching the game. It first checks if an update is available by examining the `updateText` object.
- If an update is available, it displays a message to the player indicating that they need to download the latest version.
- If no update is available, it checks if the `useSceneSwitcher` flag is enabled. If it is, it calls the `SwitchScene` method to switch to a different scene based on user selection. This is only used for an integrated Kit Patcher. See *MMORPG Integration documentation*.

- If `useSceneSwitcher` is not enabled, it constructs the full path to the game executable and launches it asynchronously using `Application.OpenURL`.
- After launching the game, it initiates a coroutine called `QuitAfterDelay` to quit the application after a short delay.

SwitchScene():

- This method is called when the `useSceneSwitcher` flag is enabled in the `LaunchGame` method. It switches the scene using `SceneManager.LoadScene`, loading the scene specified by the `sceneToSwitchTo` variable. This is only used for an integrated Kit Patcher. See *MMORPG Integration documentation*.

QuitAfterDelay():

- This coroutine quits the application after a short delay.

ServerStatus Class

Methods

Start():

- In this script, it initiates an asynchronous operation to check the server status by calling the `CheckServerStatus()` method.

CheckServerStatus():

- This method continuously checks the status of the server in a loop. It asynchronously pings the server using the `PingServer()` method and updates the UI text element (`statusText`) based on whether the server is online or offline. If an error occurs during the process, it catches the exception and displays an error message in the UI text element.

PingServer():

- Performs the ping operation to check if the server is online. It asynchronously attempts to establish a TCP connection to the server using the provided host address and port number. If the connection is successful, it returns true, indicating that the server is online; otherwise, it returns false.

UIMenuController Class

Methods

- **Start():**

- Activates the first sub-menu panel from the array of sub-menu panels (**subMenus**) by default. If the array is not empty, it sets the **activeSubMenu** reference to the first element and activates it.
- **ToggleSubMenu(int subMenuIndex):**
 - This method is a public function called when a UI button associated with a sub-menu panel is clicked. It takes an integer parameter **subMenuIndex** representing the index of the sub-menu panel to be toggled. It deactivates the currently active sub-menu panel (if any) and activates the selected sub-menu panel based on the provided index.
- **DisableLauncher():**
 - This method is responsible for disabling the launcher based on the value of the **useLauncherDisable** flag. If **useLauncherDisable** is true, it checks if the **panelToDisable** reference is assigned. If the reference is assigned (not null), it disables the UI panel by setting its **SetActive** property to false, effectively hiding it from the user when the Launch/Play button is clicked. This is only used for an integrated Kit Patcher. *See MMORPG Integration documentation.*

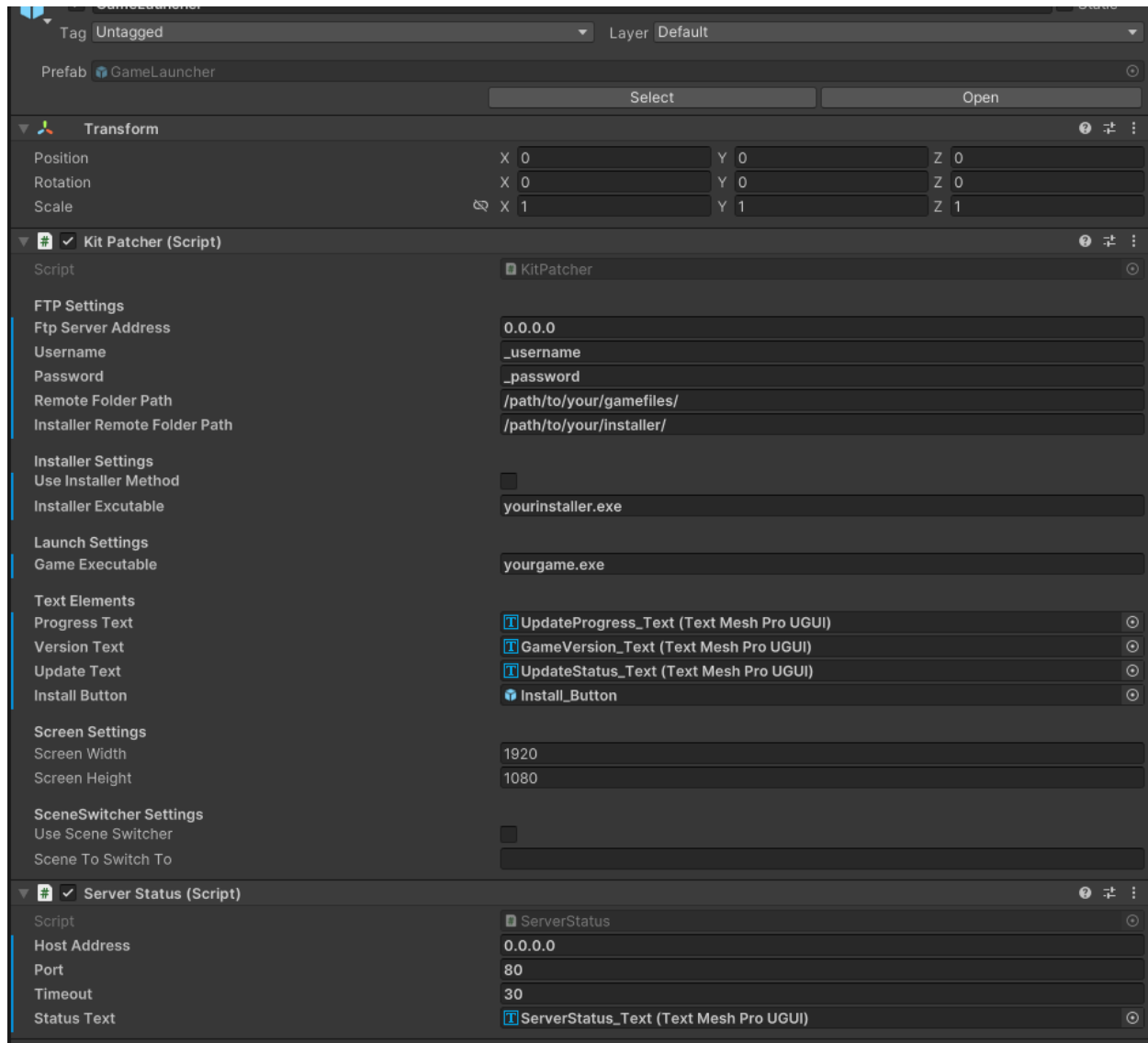
UIButtonSound Class

Methods

- **Start():**
 - Retrieves the AudioSource component attached to the button and assigns the hoverSound clip to it if both the AudioSource and hoverSound are not null.
- **OnPointerEnter(PointerEventData eventData):**
 - This method is called when the mouse pointer enters the button. It plays the hoverSound clip using the AudioSource component attached to the button if both the audioSource and hoverSound are not null.
- **OnPointerClick(PointerEventData eventData):**
 - This method is called when the button is clicked. It plays the clickSound clip using the AudioSource component attached to the button if both the audioSource and clickSound are not null.

Documentation

GameLauncher Prefab



This is where all of the patcher and FTP server setup is handled. When entering username and password, please ensure there are no spaces before or after as this will cause a login error.

KitPatcher

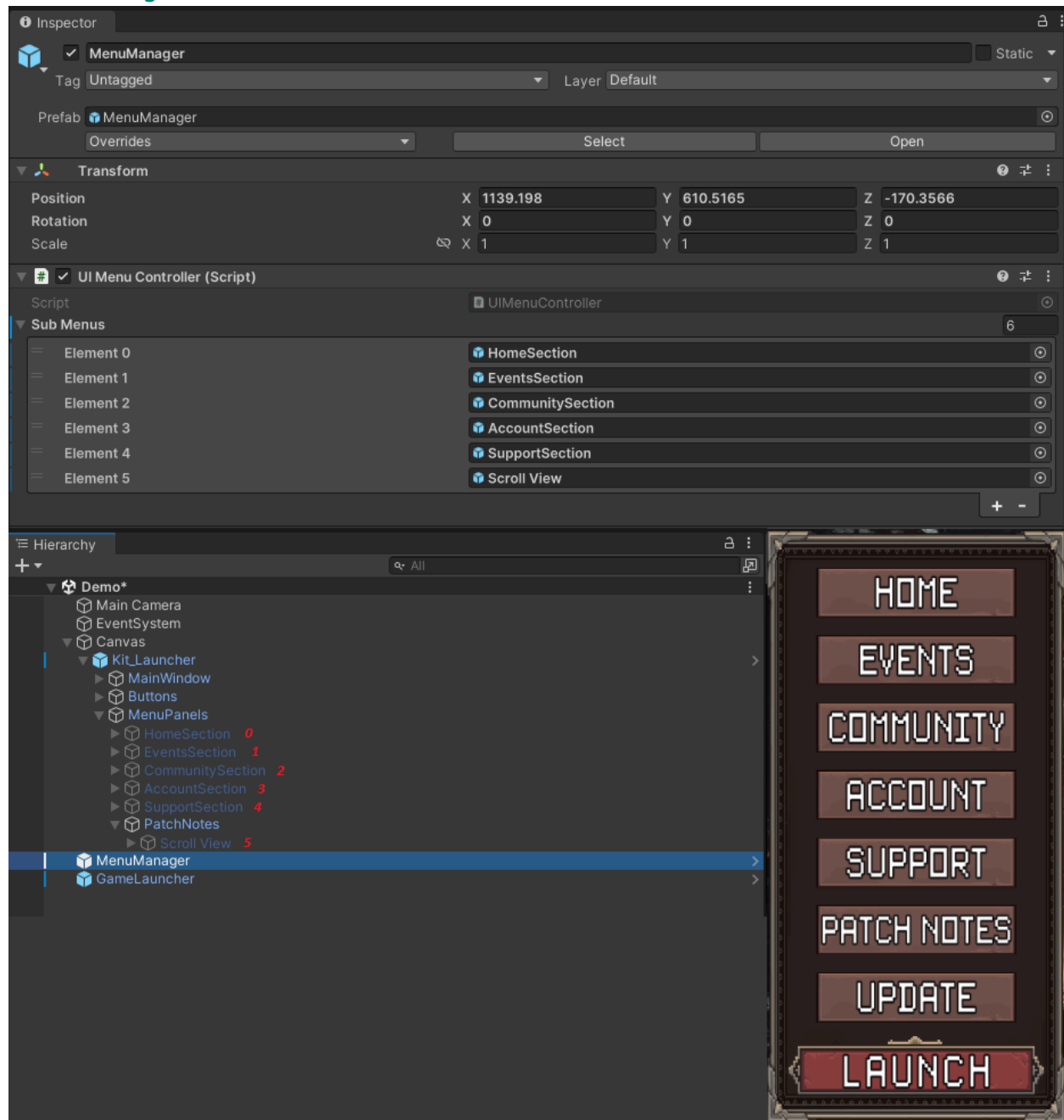
- **FTP Settings:**
 - **ftpServerAddress:** The IP address or domain of the FTP server hosting the game files
 - **username:** Username for FTP authentication
 - **password:** Password for FTP authentication
 - **remoteFolderPath:** The path to the remote folder on the FTP server. This is the folder where your game files are stored for users to download. (ex. /home/user/Desktop/file_folder)
 - **installerremoteFolderPath:** The path to the remote folder on the FTP server where your game installer package is located
- **Launch Settings:**
 - **gameExecutable:** Path to the game executable file. The .exe file of your game
- **Installer Settings:**
 - **useInstallerMethod:** Choose if using a game installer package
 - **installerExecutable:** Name of the installer package executable
- **Text Elements:**
 - **progressText:** UI text for displaying file downloading progress
 - **versionText:** UI text for displaying the local game version
 - **updateText:** UI text for displaying available game updates.
 - **installButton:** UI button for game installation
- **Screen Settings:**
 - **screenWidth:** Width of the screen resolution
 - **screenHeight:** Height of the screen resolution
- **Private Variables:**
 - **localFolderPath:** Path to the local game folder
 - **currentVersion:** Current version of the game client
 - **localFileChecksums:** Dictionary to store local file checksums
- **SceneSwitcher Settings**
 - For game integration (dedicated scene)
 - **sceneSwitcher:** Switches scenes from the launcher scene to the next scene in your game

ServerStatus

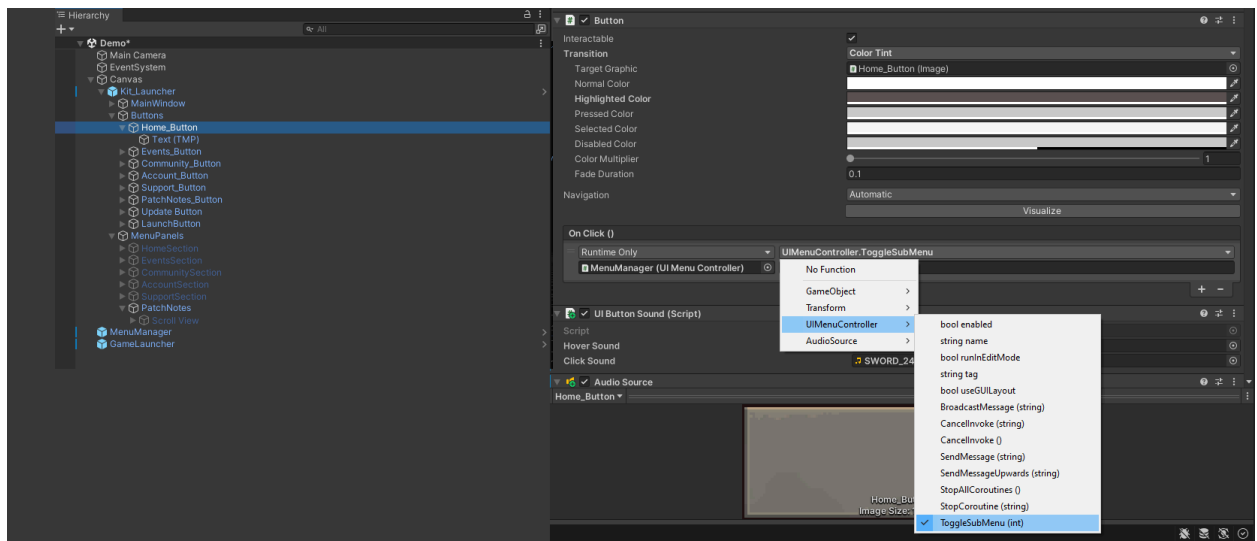
- **hostAddress:** IP address or domain name of the server to ping
- **port:** Port number to ping on the server
- **timeout:** Timeout value for the ping request in seconds
- **statusText:** Reference to the UI text element used to display the server status

The **ServerStatus** script is responsible for checking and displaying the status of a server in the launcher interface. It uses TCP/IP to ping the server and determine if it is online or offline.

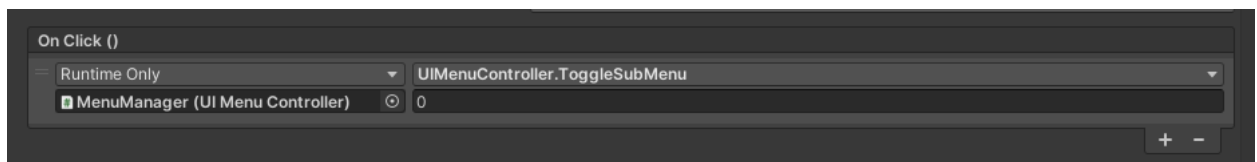
MenuManager Prefab



The **MenuManager** prefab controls the buttons and menus with the **UIMenuController** script. This is a simple way to display a UI panel with a button click. When the next button is clicked, the previous UI panel is disabled, while the new panel is enabled. Additionally, users may add more buttons and menus, even more sub-menus.



Simply add a new row element to **Sub Menus** and drag your new UI panel inside. Next select the button you wish to link to the new UI panel and add an `OnClick()` method in the Inspector. Drag on drop the **MenuManager** prefab into the `OnClick()` field, then choose the drop down box and choose **UI Menu Controller > ToggleSubMenu(int)**.



Be sure to enter the correct UI panel element number from the **MenuManager** in the `ToggleSubMenu` field. The new UI panel and button are now linked. Lastly, disable the new menu in the Hierarchy.

See video documentation.

Adding UI Button Sfx

To add a hover and click when the user uses a button, simply add the **UIButtonSound** script to the button in the Inspector window. Assign an **AudioSource** and configure it to your needs. Next, assign desired sounds to the 'hover sound' and 'click sound' fields.

FTP Server Game Files

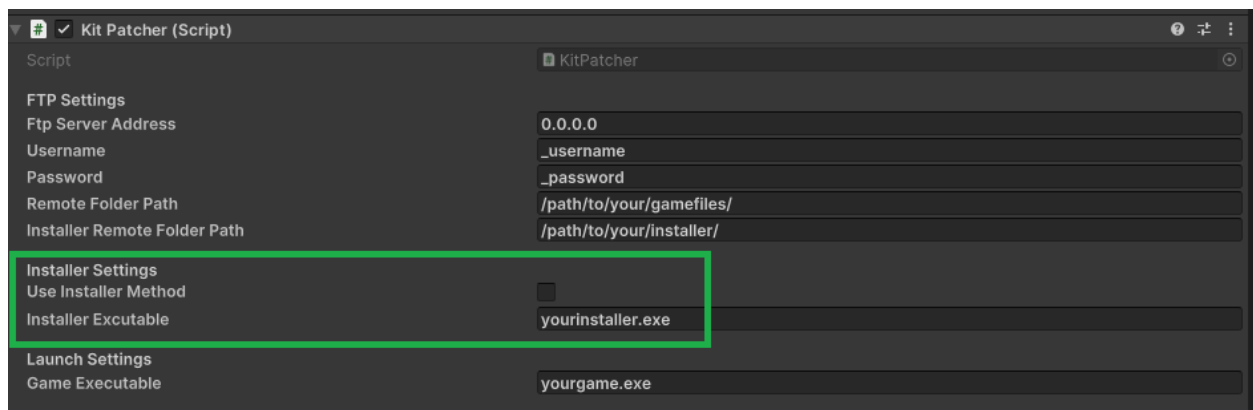
Place your Unity build game files in a folder, and place in a directory in your FTP server. Add the **Version.txt** file to the game files folder. This is the folder the launcher will download for new installations and updates.

Installer Package Executable

If you wish to provide a game installer package for new installations, chose the

Use Installer Method feature, and input the name of the executable. (ex, installer.exe)

This is a bool, which allows new installations to use the DownloadInstaller(), while updating game files uses the downloadfolder().

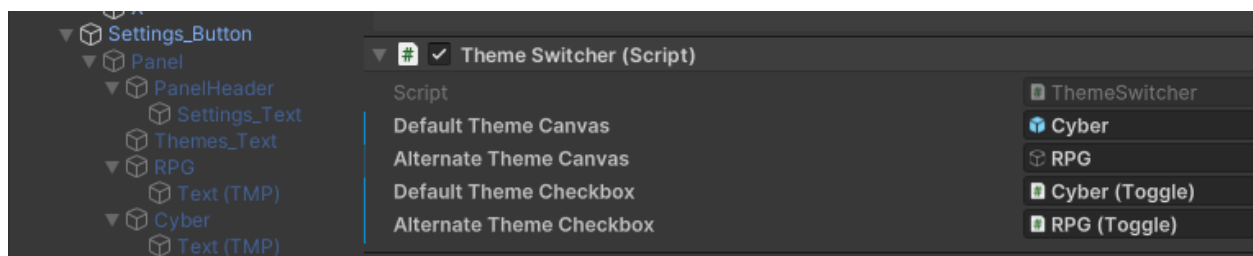


Version.txt

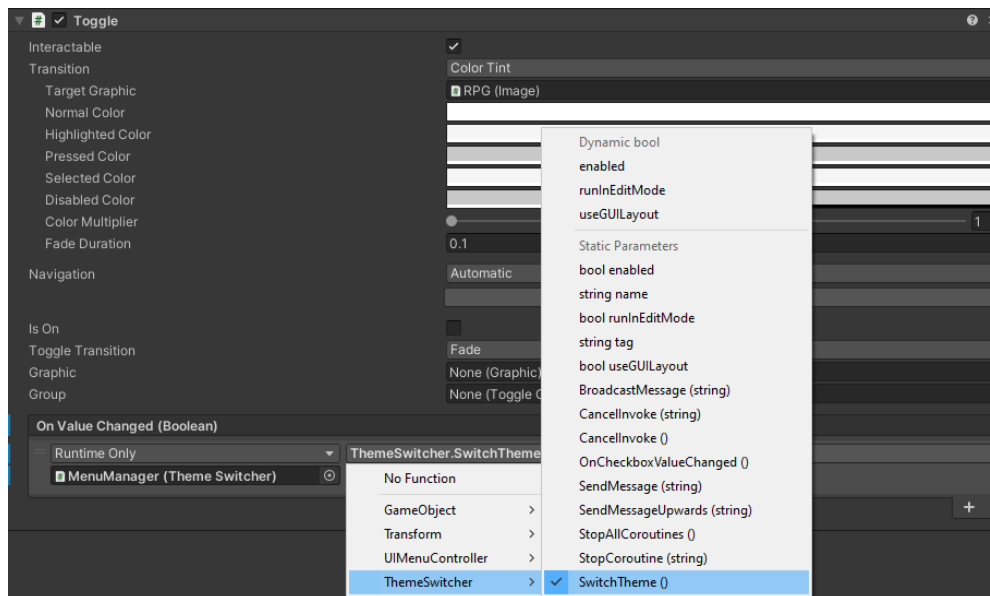
Place a txt file named 'version' in your server files folder. Inside the txt file, type your game client version. (example v0.1). This file is checked during launch and compared against the local version file.

Theme Switcher

Theme Switcher is a feature which lets developers offer an alternate launcher UI theme. Players can switch between the two themes.



Place the canvas prefabs that you wish to use as themes in the scene, and disable the alternate theme canvas. In the **MainMenu** prefab find the Theme Switcher component. Add the default and Alternate Theme canvases in the fields.



NoFunction>ThemeController>SwitchTheme()

Create a settings UI, add the needed text, toggles or buttons for the themes menu. Add your toggles or buttons to the Default and Alternate Theme Canvas fields. For each toggle/button, drag the **MenuManager** into the `OnClick()`, and select the `SwitchTheme()`.



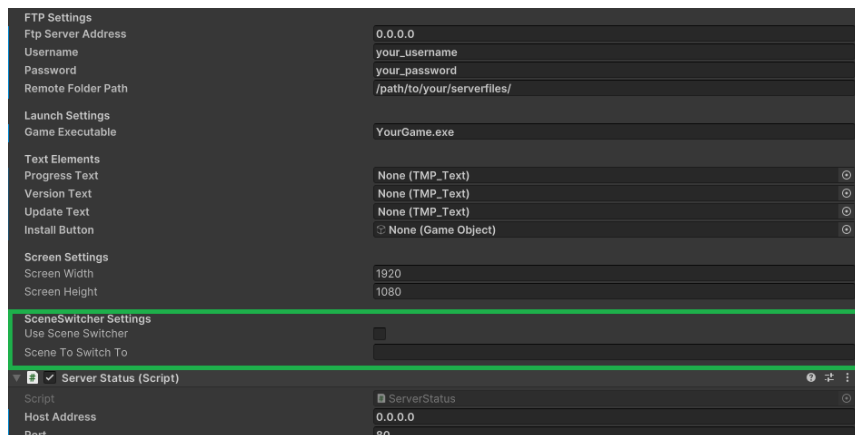
Video Documentation v0.6

<https://youtu.be/qNr4sINv56s>

Kit Integration

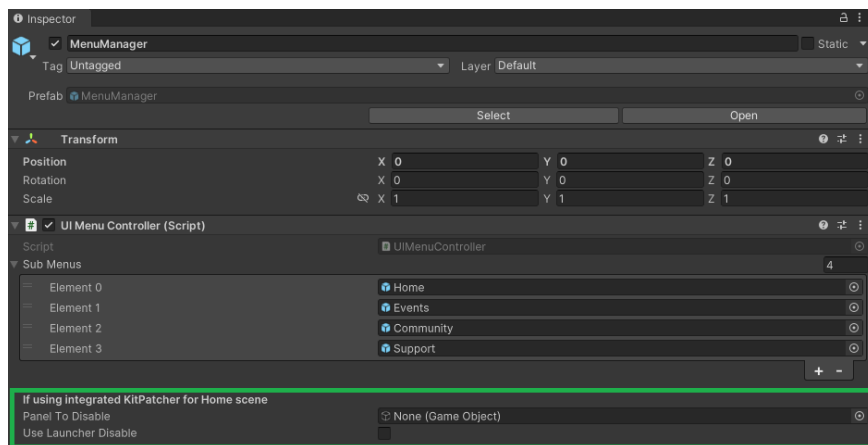
Kit Patcher works as a standalone unit or a built-in unit. There are two different methods for this. First method integrates Kit Patcher into its own scene within your game. The second method involves placing Kit Patcher into the Home scene.


Scene Switcher



Use Scene Switcher is an integration feature for using Kitpatcher in its own dedicated scene within your game. Once the player selects the play button, the launcher scene will switch to your “home” scene, or the next scene. To use, simply select the “Use Scene Switcher” checkbox, and input the name of the next scene.

Disable Launcher





Use Launcher Disable is another integration feature which allows Kitpatcher to be used in your Home, or Login scene. This option will disable the launcher window when the Play button is pressed.

See the MMORPG Kit integration documentation for an example on integrations

F.A.Q.

If you are having issues, remember to check the Debug Log

Q: I click the launch button but nothing happens and the game doesn't open

A: Is there a game update available to download? Check the notification in the top left of screen. Additionally, an update message will appear at bottom middle of screen.

*A: Make sure you have set the correct values for Sub-Folder Name, Folder Name and Game Executable Path. Ensure there are **no** spaces before or after the values. Game Executable Path should be the name of your game's .exe file. Ex. Game.exe.*

Q: My download gets interrupted with this error "Failed to download"

A: Make sure you have enough disk space to complete the download and ensure read/write permissions are granted in the game directory.

A: Check your internet connection and try downloading a different file to test.

A: Check the availability of the FTP server and ensure it is online and accepting file transfers.

Q: I moved the game directory and now my game version reads "Unknown" and "Update Available" is always showing

A: if you changed the directory of the game files, you just need to update the files and folder by clicking the Update button and choosing the new directory.

Q: Can I integrate this launcher into my Unity project?

A: Yes. Kit Patcher is designed for easy portability and functions as a stand-alone patcher or a built-in unit. Integration to the MMORPG Kit is in development now.

Version Log

v0.2

Game client files are compared to server files by last modification

Fixed screen resolution set to 800x600

Update progress now shown in launcher

V0.3

Client and server files now use version.txt file method of checking if a new update is available. Mismatched versions trigger an update download, files are checked and new files are downloaded

Implemented asynchronous file downloading to utilize concurrent file downloads

Implemented additional error handling and debug code

Added server tracking feature. Displays server status as Online or Offline

V0.4

Replaced last modified file type checking for file checksums. Now server file and client file checksums will be compared by the update check. This uses more resources but is a more precise method for checking files.

Added current game client version. This will display the current game client version on the local machine. If a game update is available, 'Update Available' will be displayed on screen.

UI overhaul with menus

Screen resolution is now resizable

V0.5

Added logic for assigning UI button sfx for hover and click

Added a quit process. When game is launched, launcher will close.

If update is available, launch game button is inoperable. Once update is installed and matching version then launch game button becomes operational.

V0.6

Partial integration for Kit Patcher into the MMORPG Kit

Can use an integrated Kit Patcher in its own scene, or add to the Home scene

Added Install button. Enables if no game folder is found. Disables when found

V0.7

New UI theme switcher lets players change the launcher layout. I tested with different themes altogether, developers can create different layouts in the same theme if desired

Themes are accessed by the settings cog

Added a settings button (cog) Added an application quit button.

Multi-threading - Downloading now starts on a sub thread, while the main thread is dedicated to the launcher

V0.8

Added Use installer package feature for first time installations.

Developers can now use a game installer package for new installations. This feature downloads and installs a Windows installer package

Roadmap

Multi-method file transfer (Google Drive, Web)

Real-time progress bars

File Decompression

Web view/HTTPS

Contact

Join our Discord for support with this asset

Ghostdog Studio

<https://discord.gg/u5b8kyWRgF>