TehModAPI

Modding API Documentation

To be completed

Contents:

Chapter 1: Installation

- Installing Visual Studio Express C# 2010
- Installing the XNA 4.0 Framework
- Downloading TehModAPI
- Setting up the basic mod Visual Studio settings

Chapter 2: Hello World!

- Making your first mod and command

Chapter 3: What is the Toolkit?

- The Toolkit explained
- Using the Toolkit within a command

Chapter 4: What is the Mod Menu?

- The Mod Menu explained
- Types of Mod Menu settings
- Setting up a Mod Menu
- Integrating a Mod Menu option into a command

~~Chapter 5: (Advanced) Drawing with the API~~ COMING SOON

~~- The basics of XNA drawing~~
~~- The Underlay and The Overlay~~
~~- Situations that call for the Underlay~~
~~- Situations that call for the Overlay~~

~~Chapter 7: (Advanced) Mouse Detection~~ COMING SOON

~~- Detecting mouse clicks~~
~~- Dragging the mouse~~

~~Chapter 6: A Compilation of useful variables and methods~~ COMING SOON

~~- A list of useful methods and what they do~~
~~- A list of useful variables and what they effect.~~

To be completed

Chapter 1: Installation

Installing Visual Studio Express C# 2010

I am not going to go very in-depth here, to obtain Visual Studio Express C# 2010 click the following link:

http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express

From there towards the middle right of the screen where there is a drop box which says ‘Select language…’, select your preferred language and click the ‘Install Now’ button. Then click on the ‘Install Visual Studio 2010 Express C# (Language)’ button which ***is underneath the big button*** and follow instructions from there until Visual Studio is installed.

Installing XNA Framework 4.0

I am not going to go very in-depth on this either. To obtain the XNA Framework 4.0 for Visual Studio click on the link below:

http://www.microsoft.com/download/en/confirmation.aspx?id=23714

After the download is finished run ‘XNAGS40_setup.exe’ and install the XNA Game Studio 4.0

Downloading TehModAPI

The same goes here once again, navigate to my post which the link is provided below, scroll to underneath the first set of screenshots (the bottom of the post) and click on the ‘Download TehModAPI for developers’ and proceed to download it. Once ‘TehModAPI.exe’ has been downloaded navigate to your Steam installation folder (default C:\Program Files (x86)\Steam\steamapps\common\terraria - for 64bit) and move/copy the executable into that folder. Now simply create a new folder and name it ‘mods’ and the API is ready to run!

NOTE All mods are downloaded as .dll files and are run from the mods folder.**

To be completed

Setting up the basic mod Visual Studio settings

Aside from fixing errors, this is one of the most complex parts of modding using this API. However follow everything step by step precisely and it will be over in no time!

Step 1

Open Visual Studio, click on 'File' in the top left and click on 'New Project…'

Step 2

In the templates menu in the center of the popup, click on 'Windows Game Library (4.0)', give your project a name and click OK

Step 3

In the Solution Explorer on the right of the screen Right Click on 'References' and click 'Add Reference…' and you will get a popup. Click on the 'Browse' tab at the top of the popup and you will see a file explorer, navigate to your Terraria folder in steam and select 'Terraria.exe', click OK.

Step 4

Repeat step 3, but this time selecting 'TehModAPI.exe' rather than 'Terraria.exe'. Click on the arrow on the left of the 'References' folder so that you may see all the references. For both TehModAPI and Terraria right click on them and select Properties, in the Properties box click on 'Copy Local' and change the dropdown menu from 'True' to 'False'

Step 5

In the Solution Explorer, Right Click on 'Class1.cs', select Rename and give it a name ending in '.cs'. Click Yes

Step 6

In the Solution Explorer, Right Click on 'ProjectName' with the windows icon to the left of it, select Properties. You will be presented with a variety of tabs with settings.

Step 7

On the initial setting tab you are presented with, under 'Game profile' select 'Use Reach to access a limited API set supported by Windows Phone, Xbox 360 and Windows.’. This allows people with lower end computers to use your mod.

Step 8

Click on the ‘Build’ tab on the left menu. Underneath the ‘Output’ section where it says ‘Output path’ click on ‘Browse…’ and navigate it to the ‘mods’ folder that you created after installing TehModAPI.

Step 9

Note This part is tricky so make sure you understand what you’re doing or this won’t work.**

Click on the ‘Build Events’ tab on the left menu. In the text area underneath ‘Post-build event command line’ write the following (or copy it)

```
cd C:\Program Files (x86)\Steam\steamapps\common\terraria
‘TehModAPI.exe’
```

Now, if the directory listed there is not the same as the location of the Terraria folder on your computer, **make sure to change it** so that it directs Visual Studio to the correct place.

Step 10

Press Control + Shift + S to save all changes made to the project. Now press F6 on your keyboard and if you’ve followed these steps correctly there should be a new dll inside your mods folder and the Terraria mod with the API should start, to regain control of Visual Studio exit Terraria. (I find the fastest way is to right click on the taskbar and close window)

Chapter 2: Hello World!

Making your first mod and command

Ok I'll revise this and write it more in-depth once things are rolling.

First in your brand new class before 'namespace <Project_Name>' add the following lines of code, this uses the references to Terraria and TehModAPI we set up earlier:

```
using Terraria;
using TerrariaAPI;
```

Now where it says 'public class <Renamed_Class>' add ': TerrariaMod' so it looks like this:

```
public class MyMod : TerrariaMod
```

Now we can start overriding methods that the API provides and will automatically execute for you while Terraria is running. For this exercise we will be overriding the '`bool SinglePlayerCommand(TehModAPI api, string cmd, string arg, string[] args)`' method which is part of TerrariaMod, so add this method into your class:

```
        public override bool SinglePlayerCommand(TehModAPI api, string cmd, string arg,
        string[] args)
        {
                // Command detection here

                return false;
        }
```

Now to explain the parameters of the function:
TehModAPI – The API / Wrapper main class, inherits Terraria.Main. So any Terraria.Main non-static variables can be accessed through this.
cmd – The first part of a command: "/testcommand arg1 arg2"
arg – All text following the initial command: "/testcommand arg1 arg2"
args – All arguments following the initial command as an array : "/testcommand arg1 arg2"

NOTE: This method must by default return false, however if one of your commands is detected MAKE SURE you return true. This tells the API to stop searching for a command because it has been detected.

To be completed

Now the way that you should detect any command using this API**(NOTE** the ‘cmd’ parameter will ALWAYS be in full lowercase, so make sure you check for lowercast only)**:

```
public override bool SinglePlayerCommand(TehModAPI api, string cmd, string arg,
string[] args)
 {
     if (cmd.Equals("helloworld!"))
     {
         return true;
     }

     return false;
 }
```

So that is how you will always base command detection. Now we can simply call ‘Main.NewText(string)’ and show the player a message! So here is our finished code: **(Sorry for the loss of depth, I’ll rewrite this when I get the time)**

```
public override bool SinglePlayerCommand(TehModAPI api, string cmd, string arg,
string[] args)
 {
     if (cmd.Equals("helloworld!"))
     {
         Main.NewText("Hello World!");
         return true;
     }

     return false;
 }
```

Congratulations, You’ve finished your first TehModAPI mod with a command!
If you are getting errors or wrote something wrong here is the finished code:
(To run the code press F6, pressing Debug will work but give an annoying popup)

To be completed

Chapter 3: What is the Toolkit

The Toolkit explained

The Toolkit is a class supplied by the API which all general methods which any mod may find useful at times are placed. It's a Utility for creating mods. Although it is a bit bare (if) the mod gets converted into a full Terraria mod from a Wrapper (If I receive permission) the Toolkit will boom with lot's of features to make things even easier. So far the Toolkit has the following methods:

```
#World Edit
DestroyAll(int startX, int startY, int repeatX, int repeatY)
DestroyTiles(int startX, int startY, int repeatX, int repeatY, Item effectOnly = null)
DestroyWalls(int startX, int startY, int repeatX, int repeatY, Item effectOnly = null)
void DestroyWires(int startX, int startY, int repeatX, int repeatY)
void PlaceTiles(int tileId, int startX, int startY, int repeatX, int repeatY, bool
replace = false, Item effectOnly = null)
void PlaceWalls(int wallId, int startX, int startY, int repeatX, int repeatY, bool,
replace = false, Item effectOnly = null)
void PlaceWire(int startX, int startY, int repeatX, int repeatY)
void RemoveWater(int startX, int startY, int repeatX, int repeatY)
#Drawing
void DrawCenteredText(SpriteFont font, string text, Vector2 position, Color color)
void DrawCenteredText(SpriteFont font, string text, Vector2 position, Color color,
float scale)
void DrawPartedTexture(SpriteBatch batch, Texture2D texture, Vector2 position,
Rectangle source, Color color)
void DrawRectWithOutline(int x, int y, int width, int height, Color outline, Color
fill)
void DrawRectOutline(int x, int y, int width, int height, Color outline)
void DrawCircleOutline(int x, int y, int radius, Color outline)
void DrawItemBox(int itemID, int x, int y, int size)
void DrawItemBox(string name, int x, int y, int size)
void DrawItemBox(Item item, int x, int y, int size)
#Getters
Item MakeNewItem(string itemname)
Item MakeNewItem(int itemID)
Player GetPlayer()
int GetPlayerID()
bool MouseLeftDown()
bool MouseRightDown()
bool IsInGame()
bool IsInventoryOpen()
```

To be completed

Using the Toolkit within a command

Example only, will write explainations in due time:

```
using Terraria;
using TerrariaAPI;

namespace NewAPIMod
{
    public class NewAPIMod : TerrariaMod
    {
        public override bool SinglePlayerCommand(TehModAPI api, string cmd, string
arg, string[] args)
        {
            Toolkit toolkit = Toolkit.GetToolkit();
            if (cmd.Equals("whatsmyname"))
            {
                Main.NewText("Your name is: " + toolkit.GetPlayer().name);
                return true;
            }

            return false;
        }

    }
```

Chapter 4: What is the Mod Menu?

The Mod Menu explained

The Mod Menu is a settings menu you can access from anywhere in Terraria whils running TehModAPI by pressing Tab. From that there is a selection of mod pages on the left side. Upon clicking one the player is presented with

Types of Mod Menu settings

There are currently 3 types of Mod Menu settings:

0 (Boolean) – Used to toggle a Boolean or 'bool' variable declared within your mod class.

1 (Integer) – Used to increase or decrease an Integer or 'int' variable declared within your mod class.

2 (Action Text) – Used to trigger the 'ModMenuAction' method that you may override in your mod class.

To be completed

Setting up a Mod Menu

Example only, will write out explaination in due time:

```
using Terraria;
using TerrariaAPI;

namespace NewAPIMod
{
    public class NewAPIMod : TerrariaMod
    {
        Toolkit toolkit;
        bool changeBoolean;
        int changeInteger;

        // This method is ran only once when the
        // API first loads up.
        public override void Initialize(TehModAPI api)
        {
            // Get the toolkit for general use
            toolkit = Toolkit.GetToolkit();

            // Register our menu in the Mod Menu
            api.RegisterMenu(this, "Mod Menu Test");

            // Add our options that we want to
            // be visible in our settings page
            api.AddOption(this, "Boolean Test", "changeBoolean", 0);
            api.AddOption(this, "Integer Test", "changeInteger", 1);
            api.AddOption(this, "ActionText Test", "actionTextTest", 2);

            // The above 3 options will now be visible
            // in the mod menu named "Mod Menu Test"

            // NOTE: Only 1 Mod Menu may be occupied per
            // mod.
        }

        public override void ModMenuAction(TehModAPI api, MenuOption opt)
        {
            if (toolkit.IsInGame())
            {
                Main.NewText("The ActionText " + opt.action + " was clicked.");
            }
        }
    }
}
```

To be completed

Integrating a Mod Menu option into a command

Example only, will write out explaination in due time:

```
using Terraria;
using TerrariaAPI;

namespace NewAPIMod
{
    public class NewAPIMod : TerrariaMod
    {
        bool changeBoolean;

        // This method is ran only once when the
        // API first loads up.
        public override void Initialize(TehModAPI api)
        {
            // Register our menu in the Mod Menu
            api.RegisterMenu(this, "Mod Menu Test");

            // Add our options that we want to
            // be visible in our settings page
            api.AddOption(this, "Command Test", "changeBoolean", 0);

            // NOTE: Only 1 Mod Menu may be occupied per
            // mod.
        }

        public override bool SinglePlayerCommand(TehModAPI api, string cmd, string
arg, string[] args)
        {
            // Our checkmenu command to see the
            // status of the menu option
            if (cmd.Equals("checkmenu"))
            {
                // Display text the player can read
                // telling him/her what the option is
                // set to.
                Main.NewText("The menu option is set to " + changeBoolean);
                return true;
            }
            return false;
        }
    }
}
```

To be completed