



SDL:Tutorials:Complete 2D Engine Input Core

From GPWiki

 **The Game Programming Wiki has moved!** 

The wiki is now hosted by GameDev.NET at wiki.gamedev.net. All gpwiki.org content has been moved to the new server.

However, [the GPWiki forums are still active!](#) Come say hello.

Contents

- [1 Input Core Overview](#)
 - [1.1 What Should a Input Core do?](#)
 - [1.2 How the Input Core is Laid Out](#)
- [2 Explanation of the Subsystems](#)
 - [2.1 eKeys](#)
 - [2.2 cInput](#)
- [3 Conclusion](#)
 - [3.1 Final Notes](#)
 - [3.2 About the Source](#)

Input Core Overview

What Should a Input Core do?

Input is perhaps the simplest of all the cores you will write for a game engine. It really depends on how complicated you wish to make it. I'm going to stick the simple version. The

functionality should tell you if a key is currently held down and for how long. Simple enough right? If you wanted you could add in more functionality. For example can think of a fighting game that has combo moves, you could in fact add this kind of functionality to the game engine but for me it is not needed.

How the Input Core is Laid Out

- eKeys - Hold all the keys on a keyboard.
- cInput - Tells us the state of the keys.

Explanation of the Subsystems

eKeys

Before you can test what keys are being pressed you need to have a definition of the keys. Since I am using SDL I will just map the SDL keys to my own names. You might think this is pointless but it makes the engine more clean and api independent like I am aiming for.

```
enum eKey
{
    KEY_A = SDLK_A,
    KEY_B = SDLK_B,
    . . . . .
    KEY_Z = SDLK_Z
};
```

Note that I left out a bunch of key to save space on the wiki. In reality you will want to map every number, letter, symbol on the keyboard to something. The only thing this doesn't take into account is distinguishing upper for lowercase. Most of the time this isn't need except when you are typing text. Our cInput class will handle upper and lower case for us.

cInput

Now that you have the basic keys defined you can design what you want your keyboard input to handle.

```

class cInput
{
    public:
        cInput();
        ~cInput();

        bool IsKeyDown(eKey Key);
        bool IsKeyDown(eKey Key, bool UpperCase);

        Uint32 TimeHeld(eKey Key);
        Uint32 TimeHeld(eKey Key, bool UpperCase);

        void Update();
};

```

As you can see we have two different version of 'IsKeyDown' and 'TimeHeld', the first ones are case independent and doesn't matter if the key is lower or uppercase. The second versions are case sensitive. 'IsKeyDown' tells if a key is being held (the time between being pressed and released). 'TimeHeld' gives a time of how long the key has being held down for. The last function 'Update' handles the all the keyboard presses since the last update, in otherwords it updates the events. Ideally 'Update' would be ran every frame.

Conclusion

Final Notes

The input core is very simple but useful part of a game engine. Add in the features that you want in and leave the ones you don't out. It might be useful to add gamepad support to the input core but I'll leave that up to you to decide.

About the Source

As with all the other tutorials you can get the whole source code [here](#).

This tutorial was written by [Seoushi](#). Do you like this tutorial? Have any questions or

comments? Let me know or you can ask the forums.

Retrieved from "http://gpwiki.org/index.php/SDL:Tutorials:Complete_2D_Engine_Input_Core"

Categories: [C and SDL](#) | [All C articles](#) | [All SDL articles](#)

-
- Content is available under [GNU Free Documentation License 1.2](#).
 - [Link to Us!](#)