The Guide to Ansel

December 5, 2018

1 Getting Started

1.1 Prerequisites

- Visual Studio 2017
- Windows 10 SDK 10.0.1734

1.2 Installing

Getting Ansel up and running is an easy process.

1. First, go to `www.github.com/maxortner01/ansel`.
2. Download the repository as a `.zip` file.
3. Extract the `.zip` to a folder on your computer.
4. Double-click on the `.sln` in the folder (assuming Visual Studio 2017 is installed)

The Solution should run the Game project without any problems. If problems should arise, be sure to update Visual Studio through the Visual Studio Installer and ensure Universal CRT is installed.

2 Making Your Own Game

2.1 Making the Project

Using Ansel and making your own game project is super easy. The `.zip` contains all of Ansel's source code and everything, but if your intent is only to *use* Ansel rather than develop on it, you don't need most of that. In fact, you can make your own solution without Ansel and it should still run if you follow the next steps.

2.2 Linking Ansel

In order to link Ansel into your project the following steps must be taken.

1. When you have created a Visual Studio Project, right-click on it and select `Properties`.
2. Then go to `Linker` and where it says `Additional Library Directories` input the directory to `bin/'Your Configuration'/Ansel` which is where `Ansel.lib` is located as well as `bin/'Your Configuration'/AnselECS` which is where `AnselECS.lib` is located.

3. Next, go `Linker->Input` and where is says `Additional Dependencies` and click the down arrow.

4. Finally, add `Ansel.lib` and `AnselECS.lib`

2.3 Including Ansel

The final step is to include the source directories for Ansel and the Component System.

1. Right-click on your project and select `Properties->C/C++`.

2. Where it says `Additional Include Directories` click the drop-down and add the folders `Ansel/src` and `AnselECS/src`.

Now Ansel is ready to go!

3 Running the Code

3.1 Entry Point

With Ansel's `include` directory included, the `.lib` file linked, and the `.dll` in the working directory, its time to start coding. Ansel is built to be easy to use and it shows in the coding process.

Ansel works by rendering classes inherited from `Ansel::Screen`. This class contains a pure virtual function (`onUpdate()`) that is to be overridden with code that runs every frame. Before this, though, C++ needs an entry point.

```
#include <Ansel.h>

#include "NewScreen.h"

int main() {
    /* ... Ansel goes here ... */
}
```

In order to run Ansel, three things are needed:

1. An `Ansel::Window` instance.

```
Ansel::Window window(1920, 1080); // Width and height of the window
```

2. A user-created `Ansel::Screen` subclass.

```
Game::NewScreen screen(&window);
```

3. Finally, a `Ansel::Engine` instance that puts everything together.

```
Ansel::Engine engine(&window, &screen);
```

Now, to run the engine, one more line is needed:

```
engine.run();
```

After this, Ansel is off, running everything including `screen.onUpdate(float timeDelta)` every frame.

3.2 Screen Subclass

This code runs Ansel, but there's nothing but a black screen! The code that runs that defines your program goes within the `onUpdate()` function of a `Ansel::Screen` subclass. To do this you must first create a class that defines the screen space which inherits from `Ansel::Screen`. `Ansel::Screen` itself has a constructor that takes a pointer to a `Ansel::Window` instance, so this must be considered when making the subclass's constructor.

```
#include <Ansel.h>

namespace Game
{
    // NewScreen inherits from Ansel::Screen
    class NewScreen : public Ansel::Screen
    {
        /* Private game variables go here... */
        unsigned int uFrame = 0;

    public:
        // NewScreen's constructor must also initialize Ansel::Screen
        NewScreen(Ansel::Window* w) : Ansel::Screen(w) {}

        void onUpdate(float timeDelta) {
            /* All event and rendering goes here... */
            uFrame++;
        }

        void onCreate()  override {
            /* ... initialization code goes here ... */
        }

        void onDestroy() override {
            /* ... destruction code goes here ... */
        }
    };
}
```

The two last methods (`onCreate() and onDestroy()`) are optional and are invoked either when Ansel recognizes everything else is done initializing or being destroyed.

4 Troubleshooting

- A lot of errors on compile with descriptions like `"cannot open source file errno.h"` or `"the global scope has no acosf"`

  Make sure Windows 10 SDK 10.0.1734 is installed.

- Low FPS

  The reason for low FPS could simply be unoptimized code, out-of-date drivers, or an older graphics card. However, another problem could be OpenGL simply not using the GPU. With Nvidia cards, to fix this, open up the Nvidia control panel. Go to `Manage 3D settings` and set "Preferred Graphics Processor" to "High-performance NVIDIA processor."