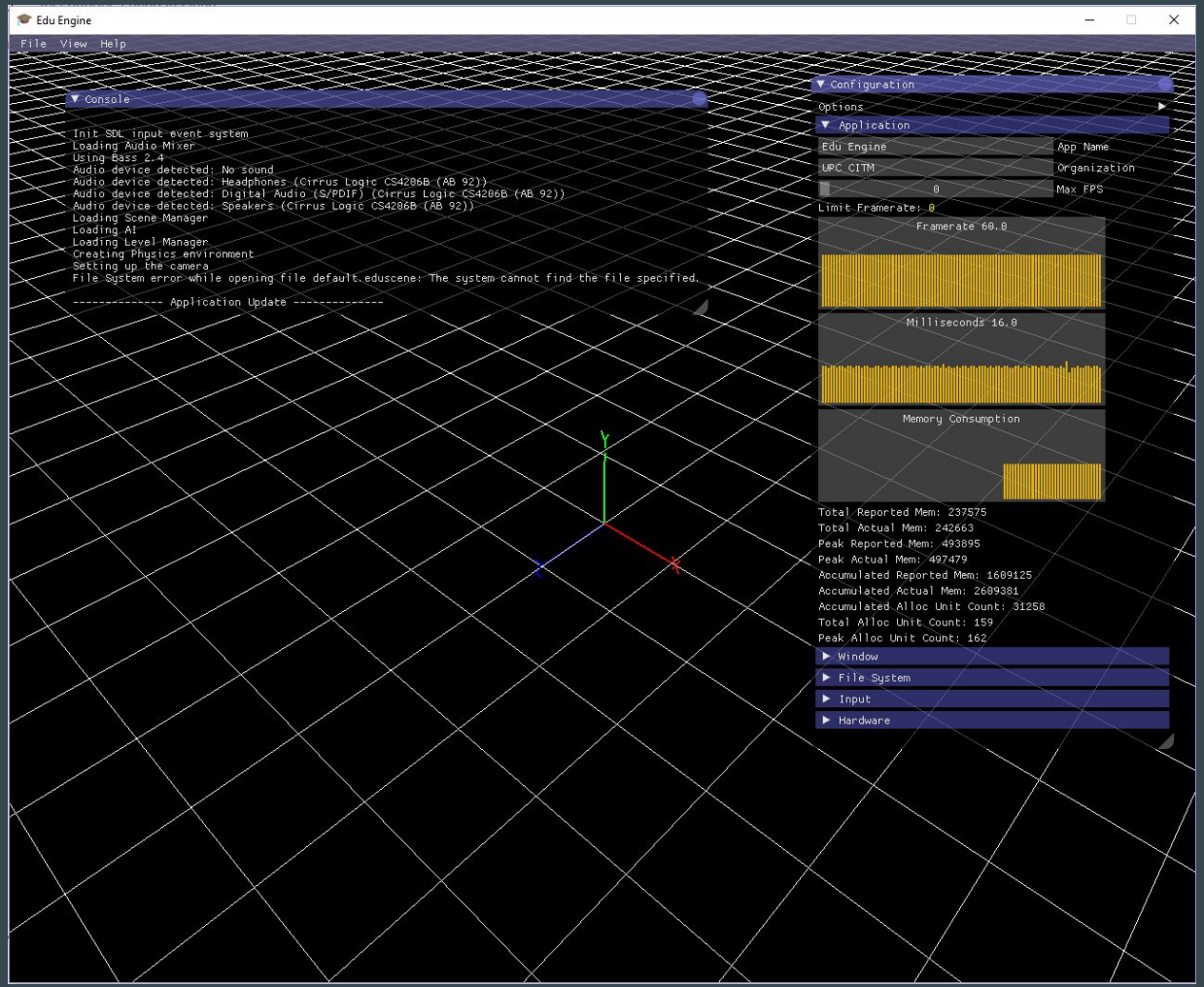
Editor Toolset

Ricard Pillosu | Marc Garrigó

Goal


Edu Engine
File View Help
Console
Init SDL input event system
Loading Audio Mixer
Using Bass 2.4
Audio device detected: No sound
Audio device detected: Headphones (Cirrus Logic CS4206B (AB 92))
Audio device detected: Digital Audio (S/PDIF) (Cirrus Logic CS4206B (AB 92))
Audio device detected: Speakers (Cirrus Logic CS4206B (AB 92))
Loading Scene Manager
Loading AI
Loading Level Manager
Creating Physics environment
Setting up the camera
File System error while opening file default.eduscene: The system cannot find the file specified.
---------- Application Update ----------
Y
X
Z
Configuration
Options
Application
Edu Engine
App Name
UPC CITM
Organization
0
Max FPS
Limit Framerate: 0
Framerate 60.0
Milliseconds 16.0
Memory Consumption
Total Reported Mem: 237575
Total Actual Mem: 242663
Peak Reported Mem: 493895
Peak Actual Mem: 497479
Accumulated Reported Mem: 1609125
Accumulated Actual Mem: 2609381
Accumulated Alloc Unit Count: 31258
Total Alloc Unit Count: 159
Peak Alloc Unit Count: 162
Window
File System
Input
Hardware

About Dear ImGui

- We will use an **immediate** mode UI known as ImGui
- Simplistic yet powerful editor widgets convenient for video game engines
  - Sponsored by Blizzard, Google, Nvidia, Ubisoft, Activision and Epic among many others
- It does not require formal installation, only to add few files to the project
- It is designed for development tools and fast iterations

About Dear ImGui - Retained vs. Immediate

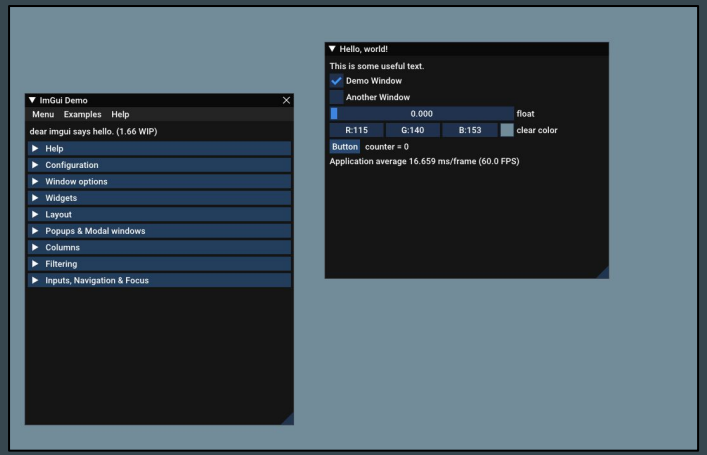
- Retained mode GUI
  - Objects are created and **retained** by the application
  - Easy to customize
  - Easy to understand and scale by the users
  - Events are handled as a callback
- Immediate mode GUI
  - Minimizes the application having to retain UI data
  - The application communicates with the API through direct function calls
  - Code tends to be more simple
  - Faster iterations during development (easier to code and debug)
  - Events are often handled in the same draw call

About Dear ImGui - Source content

- ImGui is self-contained within a few files that you should copy into your engine:
  - imconfig.h (empty by default, user-editable)
  - imgui.h / imgui.cpp
  - imgui_demo.cpp (contains a window full of examples)
  - imgui_draw.cpp
  - imgui_internal.h
  - imgui_widgets.cpp
  - imstb_rectpack.h, imstb_textedit.h, imstb_truetype.h
- ImGui is a platform-agnostic library, but we must add some “glue” code
  - We will be using SDL + OpenGL 3 implementations

Adding Dear ImGui - Sample project

- Download the project from GitHub
  - Docking branch recommended - Info about it here
- Let's begin by executing the example provided by ImGui
- Start by opening examples/example_sdl_opengl3 and compile the **project**



Adding Dear ImGui - Initial Window

- Create a folder for ImGui with the main source files (+ add them to the VS project)
- Then add the glue code:
  - imgui_impl_sdl.h / cpp + imgui_impl_opengl3.h / cpp + imgui_impl_opengl3_loader
- The main.cpp file in */examples* is a great example of integration:
  - Create a full new module for the Editor
  - Execute its init functions
    - glewInit() - (we already do it for our engine, no need to do it again)
    - ImGui::CreateContext()
    - ImGui_ImplSDL2_InitForOpenGL(window, context) + ImGui_ImplOpenGL3_Init();
  - Then its shutdown function:
    - ImGui_ImplOpenGL3_Shutdown(), ImGui_ImplSDL2_Shutdown(), ImGui::DestroyContext()

Adding Dear ImGui - Initial Window

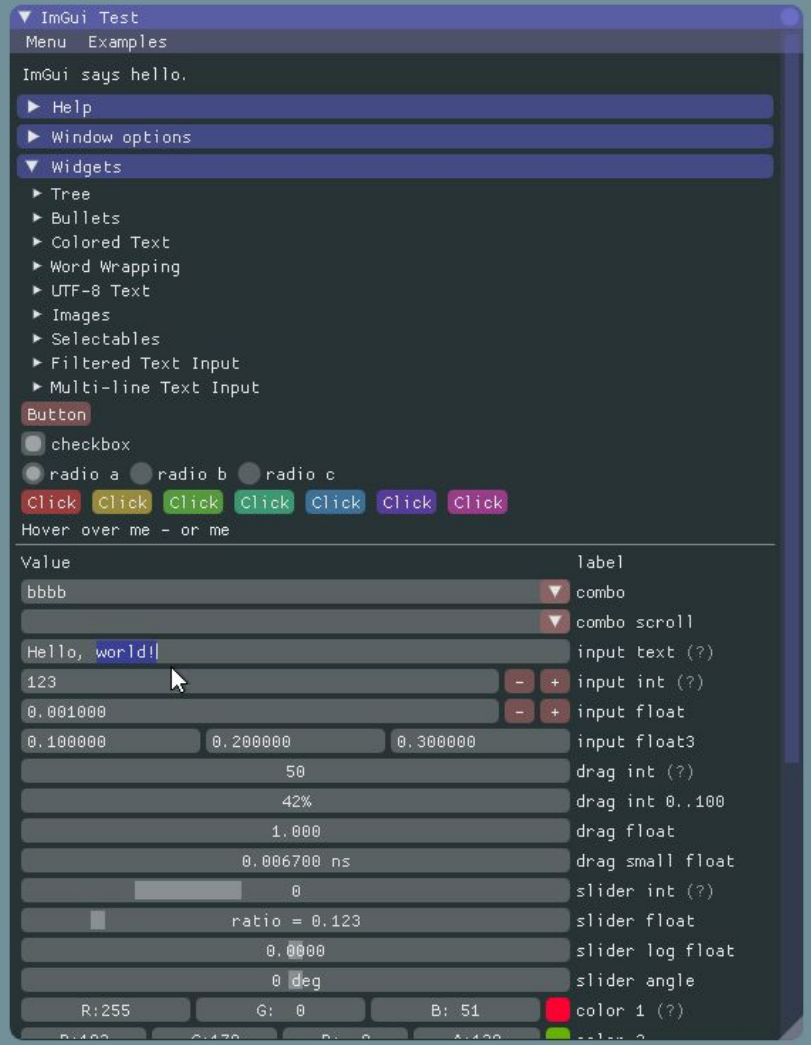
- Each loop:
  - Send input events to IMGUI calling ***ImGui_ImplSDL2_ProcessEvent(&event);***
    - We can use the input module for that
  - Start the frame with *ImplOpenGL3_NewFrame(), ImplSDL2_NewFrame() and ImGui::NewFrame()*
  - Before swapping buffer call *ImGui::Render() and ImGui's window context handling*
    - Take a look at the example!
- At this point you should have a functional setup of IMGUI
- Careful with the order of calls, use PreUpdate / Update / PostUpdate carefully
- For simplicity, it is recommended to write editor code from anywhere, not only the editor module

Using IMGUI

```
ImGui::Text("Hello, world %d", 123);
if (ImGui::Button("Save"))
    MySaveFunction();
ImGui::InputText("string", buf, IM_ARRAYSIZE(buf));
ImGui::SliderFloat("float", &f, 0.0f, 1.0f);
```

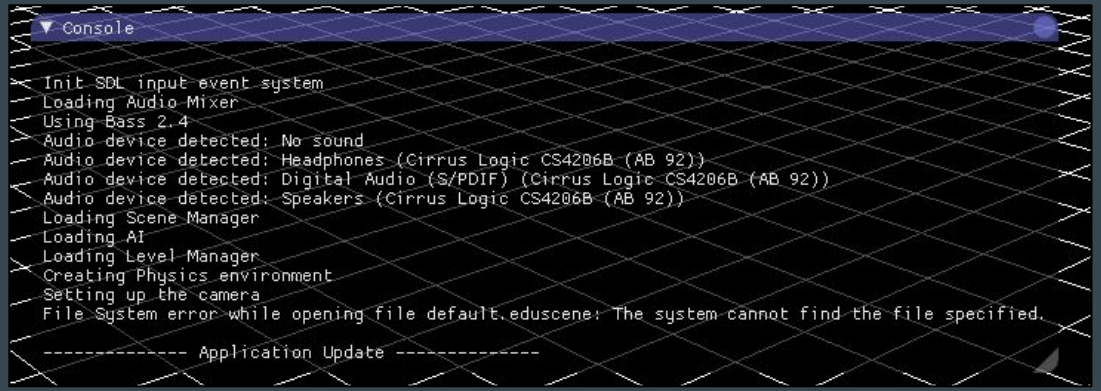
First, draw the demo window

- Simply call *ImGui::ShowDemoWindow()*
- If you want to learn how to do something with IMGUI, just read the demo code
- No … there is no formal documentation, just *imgui_demo.cpp*
- Play a bit with all the options in the demo window to understand the capabilities of IMGUI



Now let’s create a console window

- It should simply output everything that goes into LOG
- Add lines to a char* vector when receiving a LOG
- *Then draw them using ImGui::TextUnformatted();*
- *Then you can add fancy things like filters and coloring*
- Full console example here in the demo.cpp file



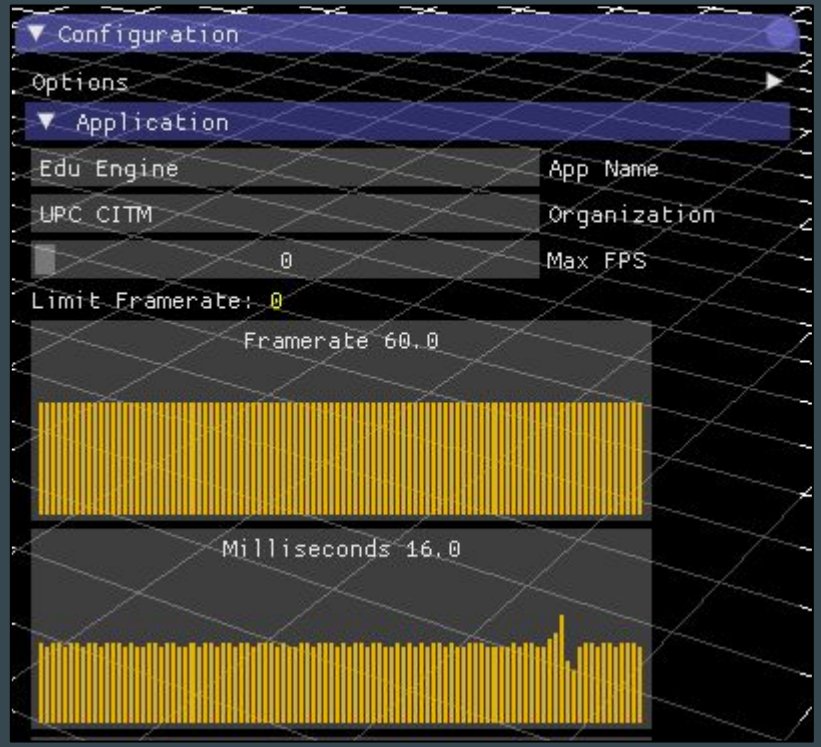
How to make a menu

- Use Module::Update
  - Or create Draw() method
- To open a browser:
  - ShellExecuteA()

```
if (ImGui::BeginMenu("Help"))
{
    if (ImGui::MenuItem("Gui Demo"))
        showcase = !showcase;

    if (ImGui::MenuItem("Documentation"))
        App->RequestBrowser("https://github.com/d0n3val/Edu-Game-Engine/wiki");

    if (ImGui::MenuItem("Download latest"))
        App->RequestBrowser("https://github.com/d0n3val/Edu-Game-Engine/releases");

    if (ImGui::MenuItem("Report a bug"))
        App->RequestBrowser("https://github.com/d0n3val/Edu-Game-Engine/issues");

    if (ImGui::MenuItem("About"))
        about->SwitchActive();

    ImGui::EndMenu();
}
ImGui::EndMainMenuBar();
```

FPS graph

- Accumulate FPS data in a vector
- When the vector is full, cycle data
- Use *ImGui::PlotHistogram() to draw*




```
char title[25];
sprintf_s(title, 25, "Framerate %.1f", fps_log[fps_log.size()-1]);
ImGui::PlotHistogram("##framerate", &fps_log[0], fps_log.size(), 0, title, 0.0f, 100.0f, ImVec2(310, 100));
sprintf_s(title, 25, "Milliseconds %0.1f", ms_log[ms_log.size()-1]);
ImGui::PlotHistogram("##milliseconds", &ms_log[0], ms_log.size(), 0, title, 0.0f, 40.0f, ImVec2(310, 100));
```

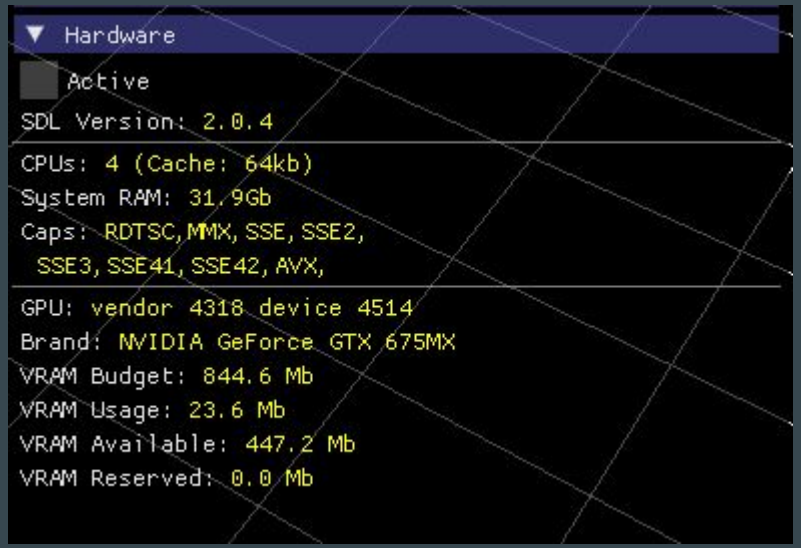
Window Options

- Expose window options
- Use ImGui::SliderInt() for numbers
- Use ImGui::CheckBox() for bools


```
if (ImGui::Checkbox("Fullscreen", &fullscreen))
    App->window->SetFullscreen(fullscreen);

ImGui::SameLine();
if (ImGui::Checkbox("Resizable", &resizable))
    App->window->SetResizable(resizable);
if (ImGui::IsItemHovered())
    ImGui::SetTooltip("Restart to apply");
```

Hardware Detection

- Either a dummy module or in Application
- Here you have some utility functions
- GPU info and memory as a bonus ;)



Homework (Nov. 15)

- Add an “About . . .” window that prints:
  - Name of your Engine
  - One line description
  - Name of the Author
  - Libraries (with versions) used
  - License
- Add some other small utilities to the editor. Look in Unity as good example.
- Explore around the docking settings and usage