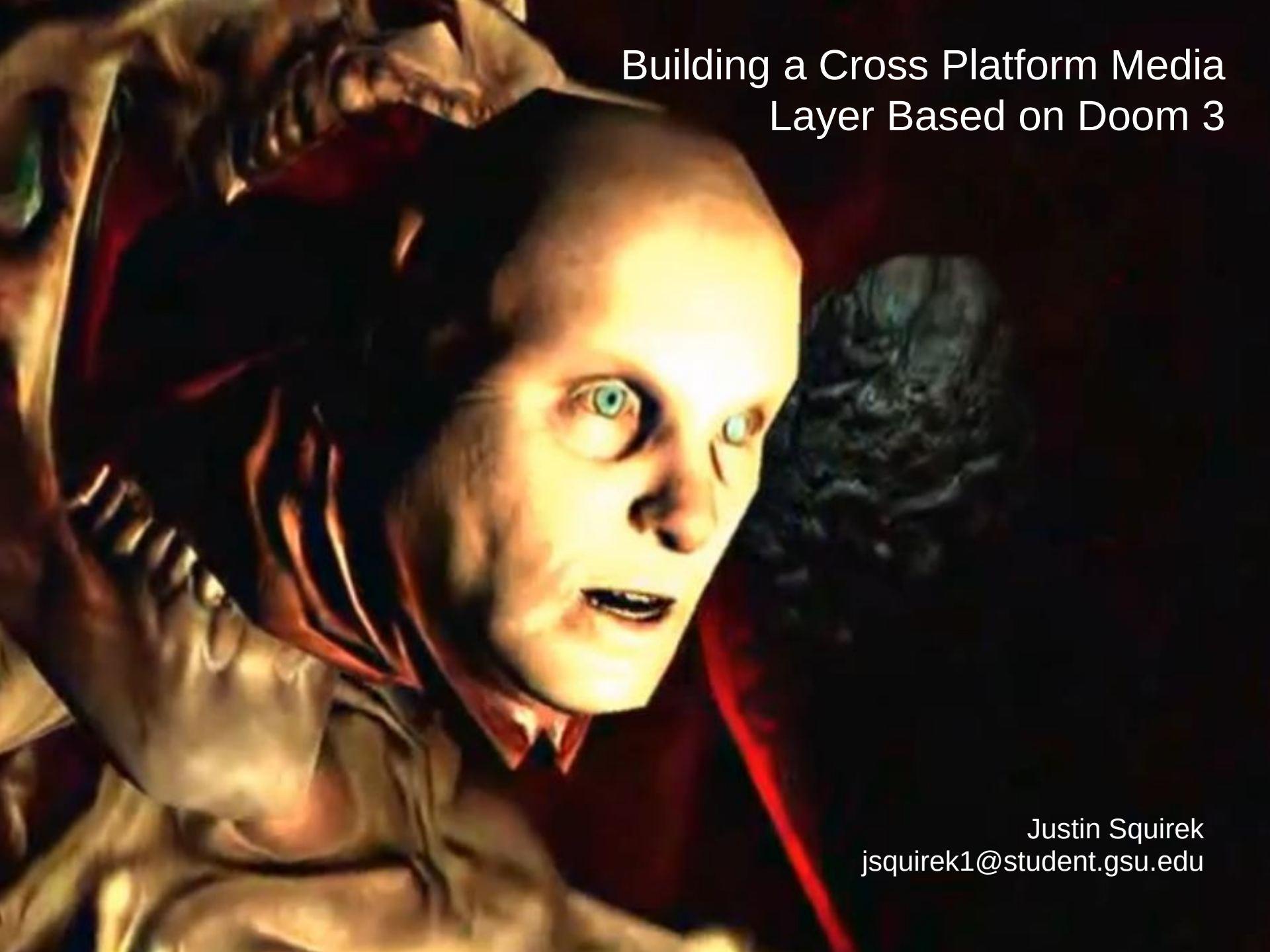


Building a Cross Platform Media Layer Based on Doom 3

Justin Squirek
jsquirek1@student.gsu.edu

Introduction

- Student
- Game developer (hobby)
- Mostly do application programming with C based languages as well as Ada and Java
- Open source developer about 2 years

What is a media layer?

- Abstract components of the system
- Ease development by the creation of a unified API
- Handle the dirty work of initializing libraries

Subsystems

- Input
- Audio
- Video
- Processor
- Networking
- Fonts and vectors?
- Image and 3D model loading?

SDL – Simple Direct Media Layer

- 15 years old
- Million of games and programs including steam
- OpenGL and DirectX
- Custom software rendering API (good thing?)
- Threads (for the C people)

Others media layers…

- FreeGLUT: Ancient, API originally designed around loss of main loop

- GLFW: OpenGL management, simple joystick support (much like SDL), keyboard, and mouse

Problems with Doom 3

- Scripting language: *“D3 was only supposed to use the interpreted code for ‘scripting’ events. It still got overused, and we did have performance issues related to it. Our takeaway was to severely deprecate its use for Rage”*

- Tasking system: *“It worked well on my dev system, but it never seemed stable enough in broad use, so we backed off from it”*

- Text based files: *“In hindsight, this was a mistake. There are benefits during development for text based formats, but it isn't worth the load time costs.”*

Designing a new media layer

- Processing of commands?
- Simple joystick support?
- Multi-monitor?
- Separation of input and windowing?
- Data file parsing?
- API extraction?

Commands and console variables

- In Doom 3 the “CVar” functionality requires components in the system layer - *but the system layer uses commands and console variable objects!*
- Event driven
- Task safe
- User input via strings without hassle

Game creation

- Not many *“AAA looking”* FPS that are open source (outside of id tech)

- Media layers ease getting started, but still only provide basic functionality

- Game engines and source developer kits remove *almost* all serious programming work

Coding

- GUI creation surprisingly huge factor
- Inverse kinematics
- Physics
- Path finding, AI events and tasks
- Animation
- Monster logic
- Experience from “AdaDoom3” ~40,000 lines

The C++ header files in Doom 3 are 447 sloc (6 more slides!). The complete implementation in Ada is 80% shorter than Doom 3's

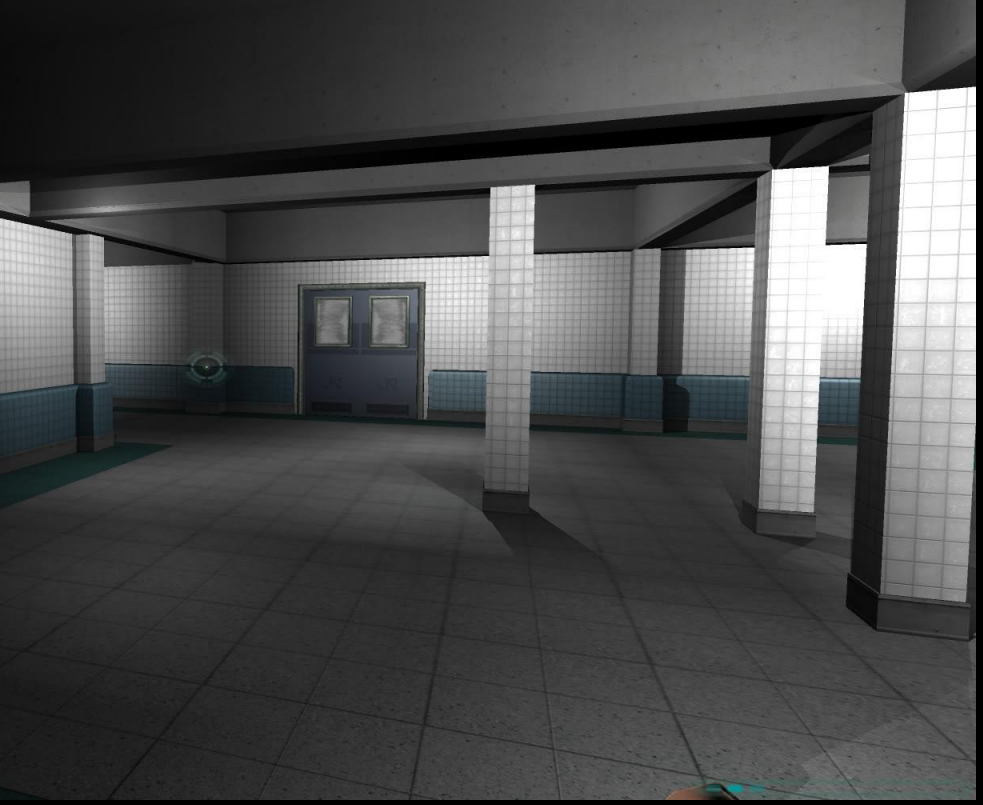
```
with Ada.Strings;
with Ada.Locales; use Ada.Locales;
with Ada.Strings.Wide_Fixed; use Ada.Strings.Wide_Fixed;
with Ada.Characters.Handling; use Ada.Characters.Handling;
with Ada.Wide_Characters.Handling; use Ada.Wide_Characters.Handling;
package Neo.Command is
   Duplicate : Exception;
   procedure Test;
   procedure Save_Variables (Path : in String_2);
   procedure Load_Variables (Path : in String_2);
   procedure Load_Language (Path : in String_2; Do_Replace_Duplicates : in Boolean := True);
   procedure Handle (Input : in String_2) with pre => Trim(Input, Both)'length > 0;
   function Autocomplete (Input : in String_2; Limit : in Integer_4_Positive := 1) return Array_String_2_Unbounded;
   generic
     Name : String_2;
     Description : String_2;
     type Type_To_Vary is (<>);
     Initial : Type_To_Vary := Type_To_Vary'first;
     Is_Saved : Boolean := True;
     Is_User_Settable : Boolean := True;
     Is_Server_Overridden : Boolean := False;
     Adjust : access procedure(Prevous, Current : in Type_To_Vary) := null;
   package Variable is
      procedure Set(Value : in Type_To_Vary);
      function Get return Type_To_Vary;
     private
      type Record_Controller is new Ada.Finalization.Controlled with null record;
      overriding procedure Initialize (Controller : in out Record_Controller);
      overriding procedure Finalize (Controller : in out Record_Controller);
      Controller : Record_Controller;
      procedure Handle_Set (Value : in String_2);
      function Handle_Get return String_2;
      protected type Protected_Type_To_Vary is
         procedure Set (Item : in Type_To_Vary);
         function Get return Type_To_Vary;
        private
         Current : Type_To_Vary := Initial;
       end Protected_Type_To_Vary;
      Data : Protected_Type_To_Vary;
   end Variable;
   generic
     Name : String_2;
     with procedure Perform(Parameters : in String_2);
   package Action is
     private
      type Record_Controller is new Ada.Finalization.Controlled with null record;
      overriding procedure Initialize (Controller : in out Record_Controller);
      overriding procedure Finalize (Controller : in out Record_Controller);
      Controller : Record_Controller;
      procedure Not_A_Formal_Subprogram(Parameters : in String_2) renames Perform;
    end Action;
private
   CURRENT_VALUE : constant String_2 := "Current value: ";
   POSSIBLE_VALUES : constant String_2 := "Possible values: ";
   INCORRECT_PARAMETER : constant String_2 := "Incorrect parameter for ";
   VARIABLE_OUT_OF_SCOPE : constant String_2 := "Variable out of scope: ";
   NO_SUCH_VARIABLE_OR_ACTION : constant String_2 := "No such variable or action!";
   MAXIMUM_POSSIBLE_VALUES_DISPLAYED : constant Integer_4_Positive := 5;
   type Access_Function_Get is access function return String_2;
   type Access_Procedure_Set is access procedure(Item : in String_2);
   type Access_Procedure_Perform is access procedure(Item : in String_2);
   type Record_Variable is record
      Saved_Value : String_2_Unbounded := NULL_STRING_2_UNBOUNDED;
      Get : Access_Function_Get := null;
      Set : Access_Procedure_Set := null;
    end record;
   package Mapped_Variables is new Maps(Record_Variable);
   package Mapped_Actions is new Maps(Access_Procedure_Perform);
   package Mapped_Names is new Maps(String_2_Unbounded);
   Names : Mapped_Names.Protected_Map;
   Actions : Mapped_Actions.Protected_Map;
   Variables : Mapped_Variables.Protected_Map;
end Neo.Command;
```

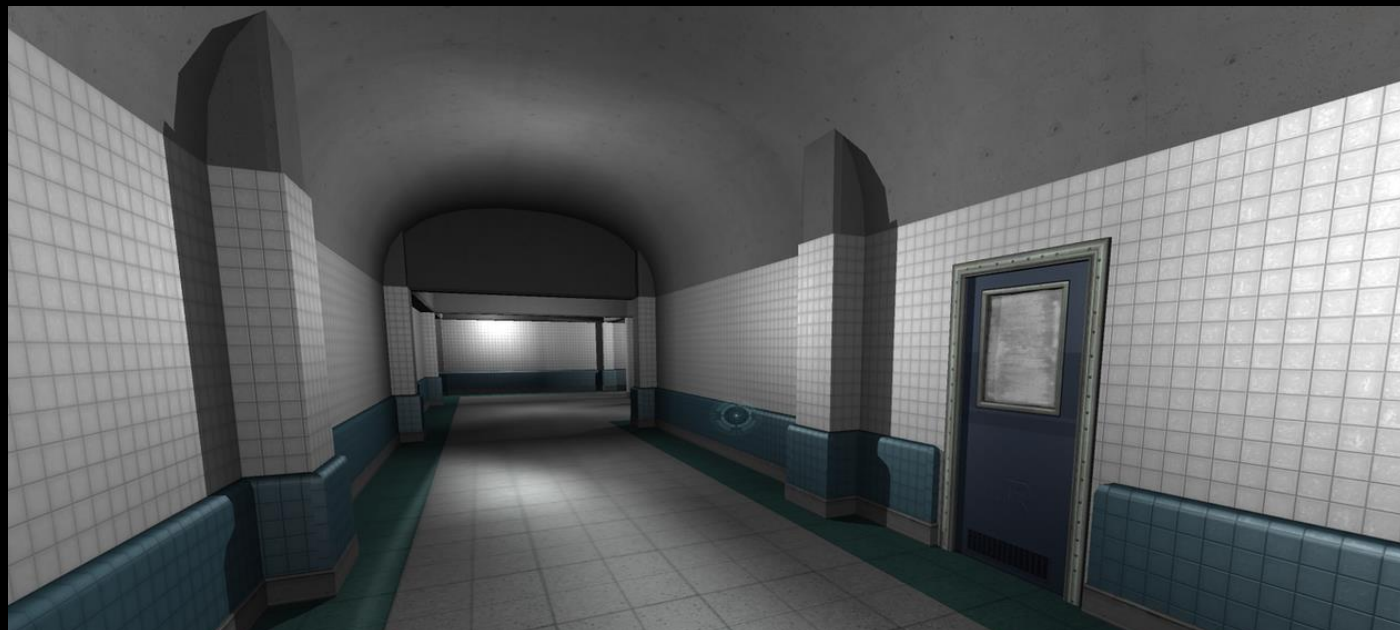
Art

- *“Technology is not all of sudden going to let you produce a AAA game”*

- Levels, interactions, and parameters are provided by engine with most being made in outside applications

- Experience from “NeotokyoMod” based on the popular half life 2 modification





DEMO OF MENU

Future of open games

- Expansion of the “The Dark Mod”?
- Development of id tech based games: Tremulous, Overdose, Xonotic, Warsow, Open Arena, Urban Terror, etc
- Many, many, more games: Super Tux Kart, V Drift, etc
- UDK draws away developers…
- Future open code releases?
- Open asset libraries?