Foundations of Game Graphics Programming

IGME 540

What is this course about?

- The basics of real-time graphics programming for games
- In other words:
  - How do *games* "do graphics"? What do *games* focus on?
- This course is not:
  - Every single graphics programming concept
  - Programmatic generation of assets or art
  - Recreating Unreal/Unity/etc.
  - Just writing shaders

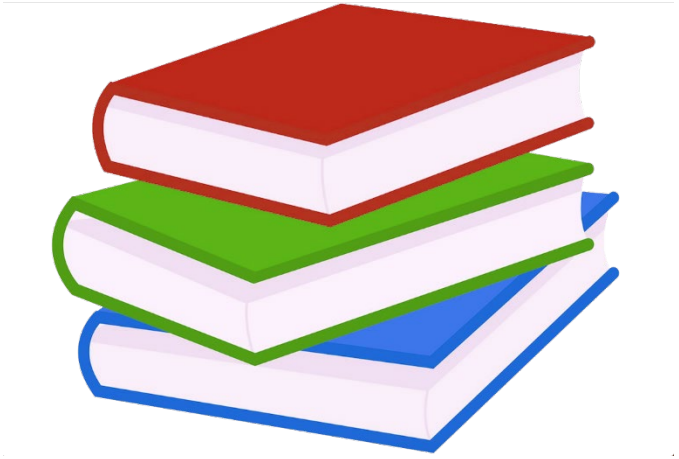
Brief course outline

- Building a basic rendering engine
  - Learning DirectX API
  - Core graphics programming concepts
  - Basics of modern rendering systems
  - Shaders: Do's & don'ts, best practices, etc.
  - General graphical techniques most games use
- Cool graphics stuff
  - I show you interesting/cool techniques
  - You create interesting/cool implementations

Prerequisite skills

- C++ competence
  - If you're not comfortable with C++
  - This is not the class for you
- 2D/3D math concepts
  - Vectors, matrices, quaternions, etc.

500-level course



- That number means something
- This is an **advanced elective**
  - You are not required to take this course
  - Make sure you're interested & committed
- You will **learn by doing**
  - Self-motivation is key

GGP work load

- Attendance & Participation
  - Up to 10% penalty for lack thereof
- Quizzes (20%)
- Individual assignments (60%)
- Final project (20%)

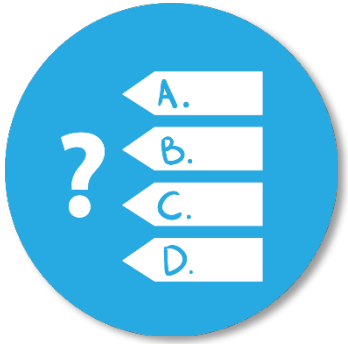
Attendance & Participation

- **Required & expected**
  - Take notes & ask questions
  - Penalty of up to 10% may be applied for poor participation
- Come prepared
  - Read through the presentation **prior** to class
  - Come with questions
- Suggest topics or specific effects
  - I'll try to incorporate them
  - Send me images/video of games or effects

Lectures & demos

- Some topics include both lectures and demos
- All of my demos are available on GitHub
- Please **do** use the code as a reference!
- But do ***not*** just copy/paste it
  - If you just hand my own code back to me…
  - You get a zero for the assignment

Quizzes



- Each major topic has a quiz
- Quizzes cover:
  - Assignments
  - Slides
  - Lectures
- Quizzes will be taken outside of class
- Another reason to take notes

Individual assignments

- Guide you through building a basic engine
- Begins with starter code
- Basics:
  - Lit, texture-mapped 3D objects
  - Basic material system
  - Free-roaming debug camera

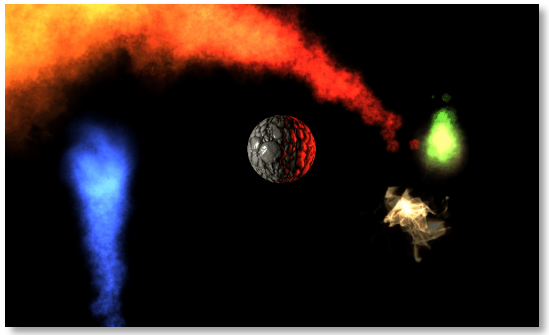
- Advanced: Normal mapping, sky boxes, PBR, etc.

Starter code

- I provide starter code for you
- Handles window creation & DirectX init
  - Like a DirectX version of GLUT/GLFW
- Draws a single triangle on the screen

Final project options

- 1. Implement an **advanced graphics technique**
  - Something we cover near the end
  - Or something you research on your own



- 2. Create a **simple game-like experience**
  - Using your engine in a game-like context
  - Think *Flappy Bird* or *Pong*

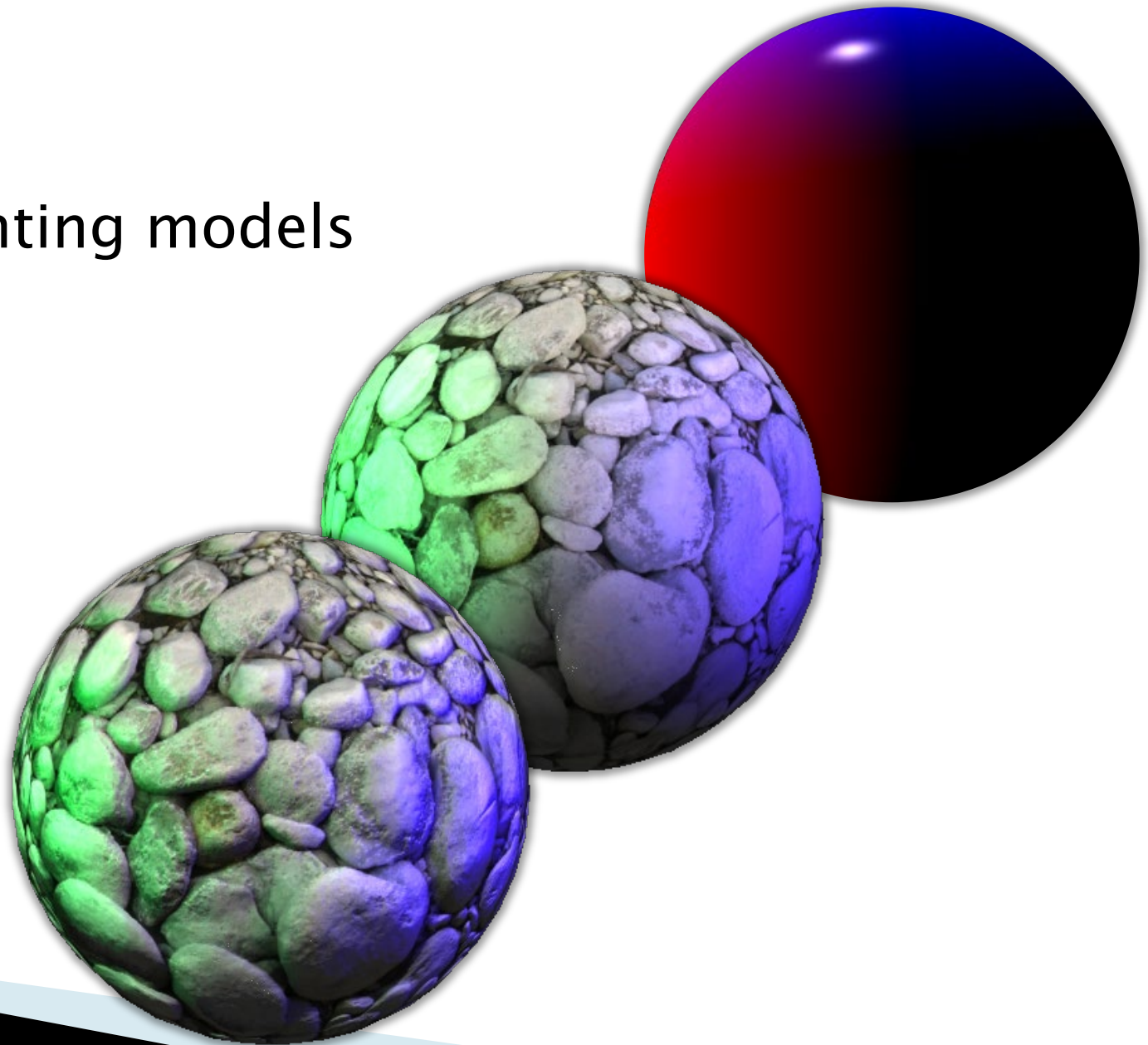
- Collaboration options
  - Solo (1 person)
  - Small group (2-ish people with 2x expectations)

Early topics

- Starter code
- Direct3D and its API
- Rendering pipeline
- DirectXMath library
- Shaders & HLSL

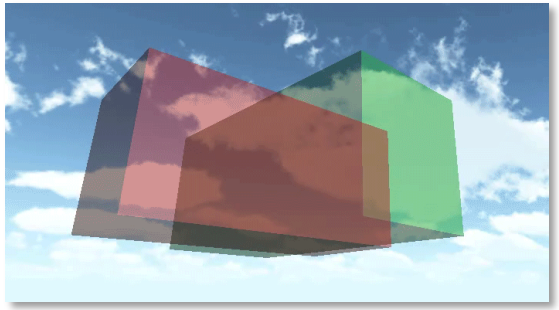
Later topics

- Basic materials & lighting models
- Texturing
- Normal mapping

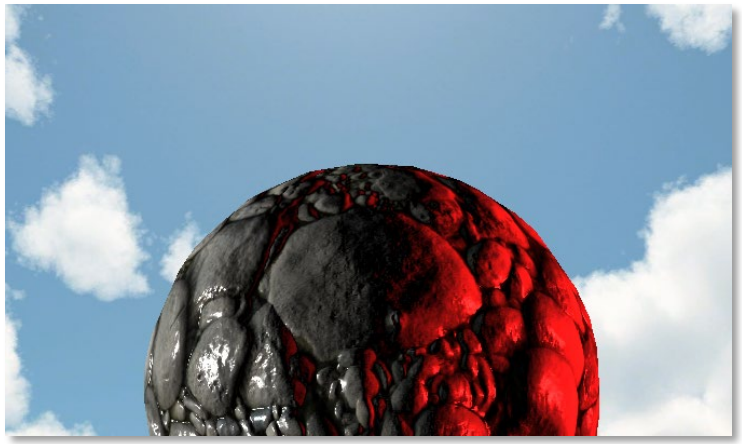


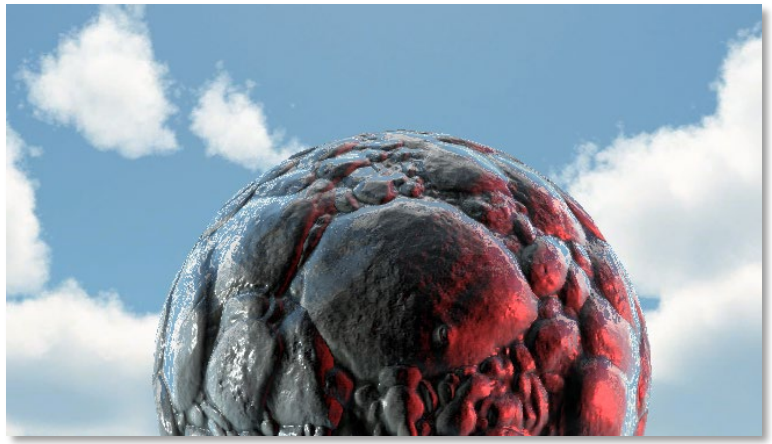
Later topics

- Blending (Transparency)



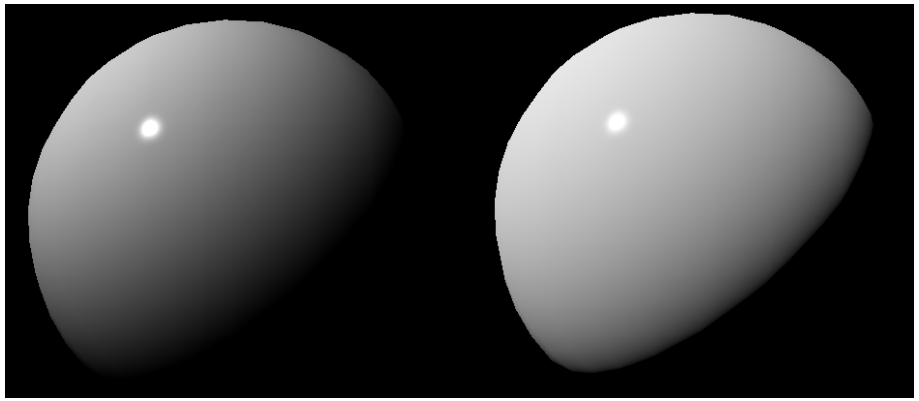
- Cube maps: Skyboxes & Reflections



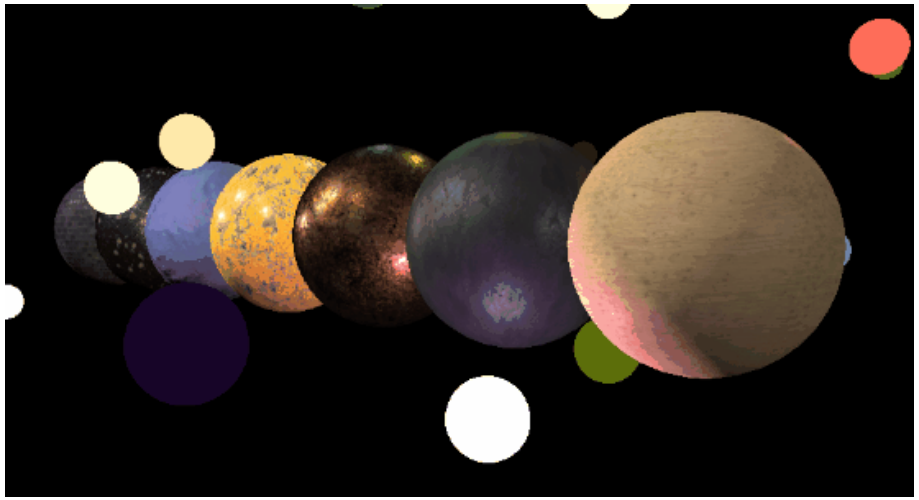


Later topics

- Advanced lighting
  - Gamma-correctness



- Advanced lighting
  - Physically-based rendering for direct lighting

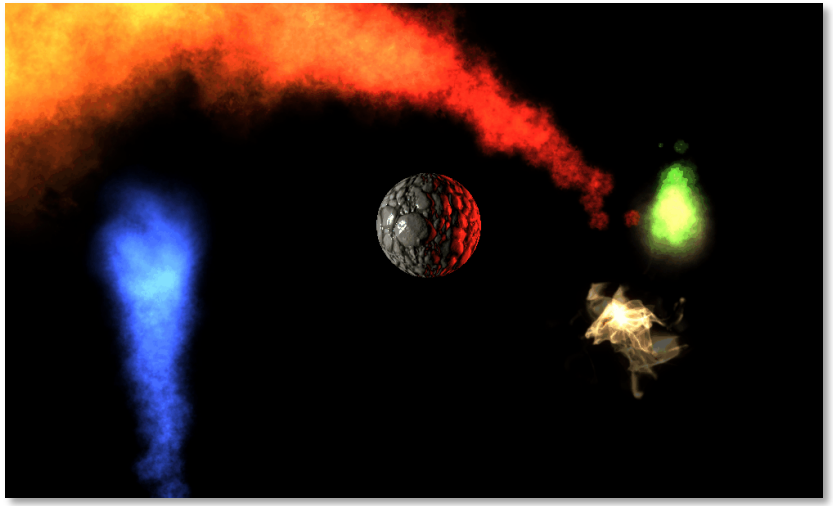


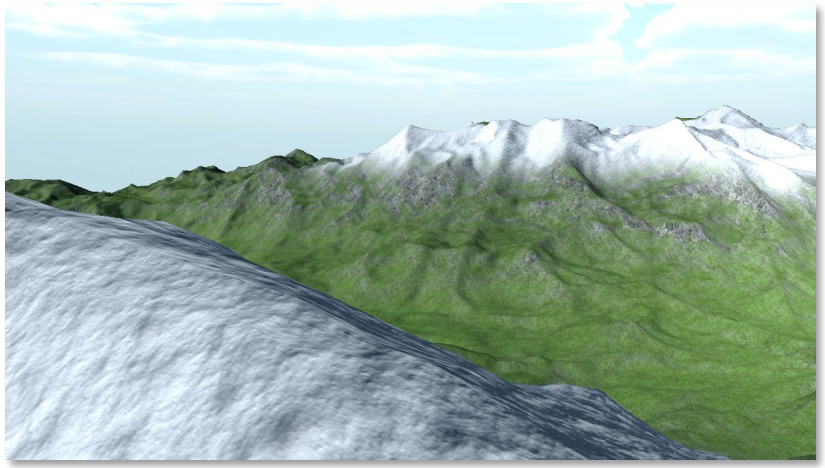
Later topics

- Real-time shadows
- Post processing
  - Render targets
  - Bloom
- Quick topics
  - Toon shading
  - Fog

Other potential topics if we have time!

- Particle systems
- Height-map terrain
- Basic water systems
- Compute shaders
- Refraction effects
- Deferred lighting





Game demo day

- I bring in game console(s): Xbox 360, PS4 and/or PS5
- We analyze games, including
  - GTA V
  - Infamous Second Son
  - Spider-Man
  - (And more)
- Near the end of the semester (usually)

Coding expectations

- Many early assignments are step-by-step
- Later topics & assignments are less so
  - We'll cover concepts
  - Overviews of implementation details
  - Potentially, examples of tricky bits of code
- It is your job to take those **concepts** and turn them into **concrete implementations**

The use of Google

- Just so we're clear…
- **Googling for code is NOT an acceptable way of completing assignments in this course**
- Can you use Google? Of course.
  - For clarifications, language syntax, API reference
  - But not "to get the code because the professor didn't give it to us"

“But what do I type?”

- I don’t expect that answer to magically materialize in your brain during class

- But by this point in your academic & programming careers, you should be able to:
  - Learn about a concept
  - Distill it down into its basic components
  - Plan an implementation
  - Write the code
  - Test, find bugs, fix, repeat

Assignment #1 – Starter Code

- 1. Download (or fork on GitHub) and run
- 2. Read document about starter code architecture
- 3. Survey
  - Did the code work on your machine?
  - Familiarity with concepts
  - Optionally, ask questions about the code base
- Note: There will be a quiz on this!