Introduction to game programming

Autumn 2011

Juha Vihavainen
Department of Computer Science
University of Helsinki

On classroom etiquette

- Please, silence your cell phone while in class.
- Please, do not take or make phone calls in class.
- Please, do not keep up discussions among each other during lectures
  - really, the lectures are voluntary and ongoing discussions disturb others wishing to listen
  - but, of course, do ask questions . .

Course outline

Overview and introduction (this lecture)

C# and .NET architecture

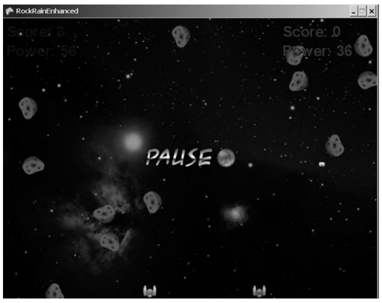
XNA game framework and game programming basics

2D games with XNA

- predefined game skeleton, components, and 2D sprites
- XNA provides support for 2D images (textures), IO, and audio

3D games with XNA

- 3D scene rendered on a 2D screen: use of graphics pipeline
- setting camera and view frustum
- using 3D models, and shaders (GPU programs)



Course goals

- understand the inner architecture of games
  - the skills necessary to participate *as a programmer* in a video game team: programming techniques, algorithms, graphics, some game-related mathematics, etc.
  - main game loop, game components, assets (resources)
  - drawing of 2D sprites and 3D objects
  - graphic pipeline, models, shader effects, audio, etc.
- develop general increased proficiency in programming
  - practice use of an OO library/framework (XNA & .NET)
  - learn a new language (C#) and its programming environment (Visual Studio)
- emphasis on game implementation issues, and not on *game design* (game's quality from a player perspective)

Working in this course

- Projects will be major element of the course
- There will be both individual and group assignments and reports
- Individual work: personal course log, active attendance in exercises, and peer evaluation and feedback on prototype games
- We will utilize XNA
  - XNA is an environment suitable for teaching the fundamentals of game development
  - C# is relatively easy to work in (for anyone who knows Java)
  - serves for both coding and scripting (in this course)
  - XNA means: "*XNA is Not an Acronym"* (note the *X*, anyway)
- Hopefully, the course is fun and an rewarding!

Course prerequisites

- general maturity in programming
  - knowledge of Java and basic object-oriented concepts, data structures (lists, arrays)
  - some programming experience
    - the more, the better . .
- no previous knowledge about graphics programming is required (but doesn't hurt, of course)
- no knowledge of .NET or C# is required
  - introduction to the relevant features of C# and .NET are given as the first part of the course

Related courses in our department

- *Game Engine Architecture*
  - actually, serves as a "*Game programming* II"
  - requires basics of 3D and some programming skills in C++
- *Software Design* (C++)
  - C++ programming language and techniques
  - framework and library design
- *Compiler Design* & *Implementation*
  - how to implement programming languages, or generally any analyzers/processors for structured texts
    - e.g. processing of scripts, configuration data, etc.

Course textbook

Tom Miller, Dean Johnson:

XNA Game Studio 4.0 *Programming - Developing for*
Windows Phone 7 *and Xbox* 360. Addison-Wesley, 2011.

http://www.informit.com/store/product.aspx?isbn=0672333457

- Authored by Microsoft XNA development team members
  - offers "insider insights"
  - technical tutorial and reference
    - only partly covered in this course
  - the book presumes knowledge of .NET and C#

Other sources

- see course web pages . .

Beginning game programming

- Games are complex applications to build
  - multiple interactions within the program
    - game entities move and change and act upon each other
  - use of visual/audio data, GPU programs, other resources
  - must interface with libraries that provide hardware services and components (especially for graphics)
- Analysis of game program architecture and concepts
  - how to design and implement a game program/engine?
  - what are relevant techniques and patterns?

XNA vs. the "real world"

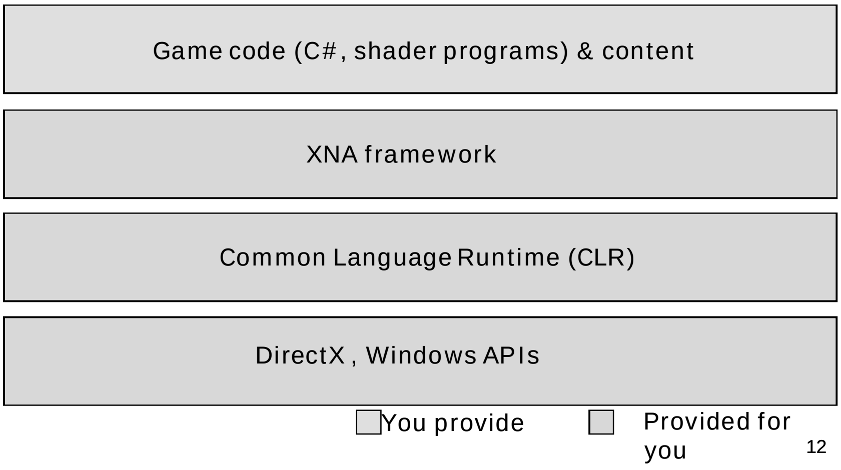
- The majority of game development is done in C/C++ and scripting languages, for example
  - **Ogre** graphics engine (over 1,300,000 lines of C++)
  - **Unreal Engine** uses C++ (for core code) and **UnrealScript** (for gameplay)
  - **OpenGL** and **DirectX** graphics libraries (low level)
- But, the concepts are mostly the same, we’re just simplifying things for educational purposes
- We’ll do little or no 3D modeling; it is assumed artists will do that for us
- We’ll only be interested in interactive video games, and not in general graphics

XNA Game Studio

- free tool for students and hobbyists (need to register)
- a set of libraries for creating 2D and 3D computer games
- uses C# as the programming language and integrated with Visual Studio C#
  - either Express or Full version of Visual Studio
  - need Visual Studio 2010 for XNA version 4.0
- games can be freely run on Windows PCs
  - *Xbox* 360 and *Windows Phone* 7 require arrangements/costs
- but XNA
  - is not quite as powerful as some C++ graphics engines (?)
  - is it really so very easy? games are still pretty complicated..

Architecture of XNA Game Studio

- write games in C# using classes in XNA library
- runs on top of CLR = virtual machine



Game code (C#, shader programs) & content

XNA framework

Common Language Runtime (CLR)

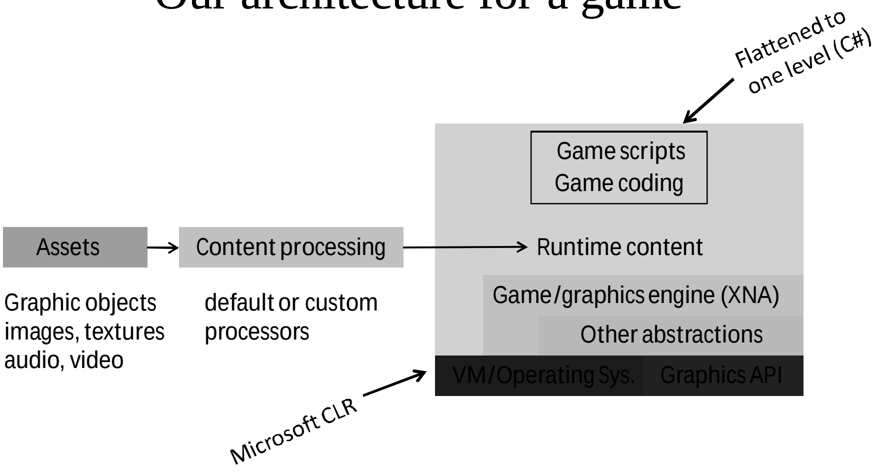
DirectX, Windows APIs

□ You provide ▨ Provided for you

XNA features

- 2D and 3D graphics (can be used separately or together)
- keyboard, mouse, Xbox 360 controller input, etc.
- a special XNA *content pipeline*
  - to process and transform different file formats into internal binary XNA format (.bna) ready for loading
  - a game gets its assets as instances of XNA classes
- audio support
  - *XACT* is a general cross-platform audio tool
- access to *HLSL* (High Level Shader Language)
  - to program vertex and pixel shaders (running on GPU)
- font support, networking, game save, and more

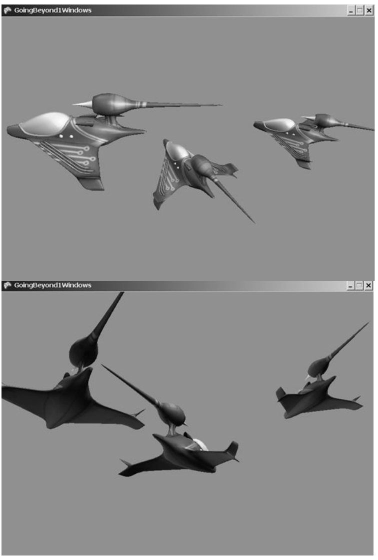
Our architecture for a game



XNA code sample

rotating 3D objects & moving camera

```
public class Game1 : Microsoft.Xna.Framework.Game {
   GraphicsDeviceManager graphics;
   public Matrix  CameraView  { get; set; }
   public Matrix  CameraProjection { get; set; }
   public Vector3 CameraPosition;  . . .
   public Game1 ()  {
      graphics = new GraphicsDeviceManager (this);
      Content.RootDirectory = "Content";
   }
   protected override void Initialize ()  {
      CameraPosition = new Vector3 (0.0f, 0.0f, 5000.0f); . .
      CameraProjection =
         Matrix.CreatePerspectiveFieldOfView (
            MathHelper.ToRadians (45.0f), AspectRatio,
            1.0f, float.MaxValue);
      new Ship {
         Position = new Vector3 (2000, 500, 50),
         Rotation = MathHelper.ToRadians (40f),
         Scale = .75f };     . . .
      base.Initialize ();
   }
   protected override void Update (GameTime gameTime)  {
      if (Keyboard.GetState ().IsKeyDown (Keys.Up))
         CameraPosition.Y +=
                  (float)gameTime.ElapsedGameTime.TotalMilliseconds * 5;
       . . .
      base.Update(gameTime);
   }  . . .
}
```



Lectures and projects

1. .NET, C#, and XNA Game Studio taught in class/exercises
   - 2D and 3D games are possible with this environment
   - created games can, in principle, run on *Xbox* 360 and *Windows Phone* 7
     - but these aspects are not (really) taught in this course
     - we mostly stick to the general concepts/techniques
   - small programming assignments as part of exercises

2. Work in teams of five (5) students to create a computer game
   - organized in self-managed teams
   - game concept and user manual
   - technical design and its document
   - first a demo version for peer testing, then a final game

Grading

- no examinations (so relax) but we do have lectures/exercises
- individual and team assignments and reports
- exercises: 6 points
- personal log: 10 points
- peer review & feedback of demo games: 8 points
- team project: 36 points
  - game concept and user manual: 6 points
  - technical design document: 8 points
  - fun, originality, and effects: 6 points
  - quality of the final software: 16 points
- maximum 60 course points

Keeping a personal log

- must start a log that you will keep updating and handing in throughout the course (dates will be given later)
- record what you did for the course since the last log:
  - what you read, what you learned (personal views), what you don’t get yet, what work and how many hours did you do, how the assignments worked out . .
- the log should be organized into reverse-ordered sections
  - don't throw away previous parts but just add a new section at top of the whole document (makes the last part easy to find)
- name the log with your personal unique login name (<*login_name*>.pdf or <*login_name*>.html)
- the log will be graded on its content, effort, readability, and the quality of presentation and originality (don't just copy lectures..)

Installing XNA Game Studio Express

- get or install Visual Studio and XNA Game Studio
  - "Windows Phone Software Development Kit (SDK)"
    - XNA 4.0 is integrated into this SDK
    - Windows and Xbox projects come along with the tools
    - downloads "Visual C# 2010 Express" as needed
  - XNA will also work with Visual Studio 2010 Professional if you have that already installed
- follow instructions on pages: "http://create.msdn.com/"
  - the current available XNA version is 4.0 (published Sep 16, 2010) - you need to stick to this version for this course!
  - the version 4.0 is covered by the course textbook

Home work: get started

- Visit Microsoft developer website
  - http://create.msdn.com/
- If necessary, download and install
  - Visual Studio C# 2010 Express and XNA Game Studio 4.0 (the only version allowed)
- Visual Studio and and XNA are also be available at the CS department computers (the room B221)
- Start compiling and playing around with sample programs
- Start thinking what (kind of) game you want to program