Game Development: Harder Than You Think

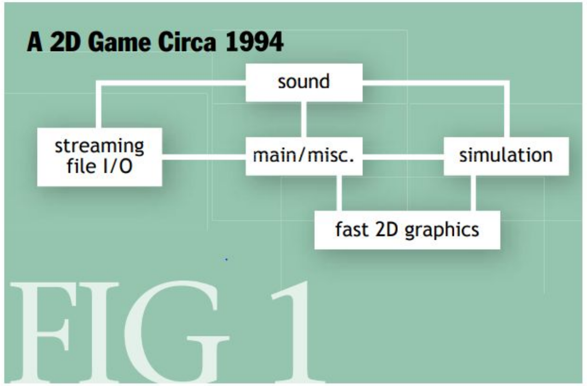
Presented by Kim Long Vu and Sondre Slåttedal Havellen

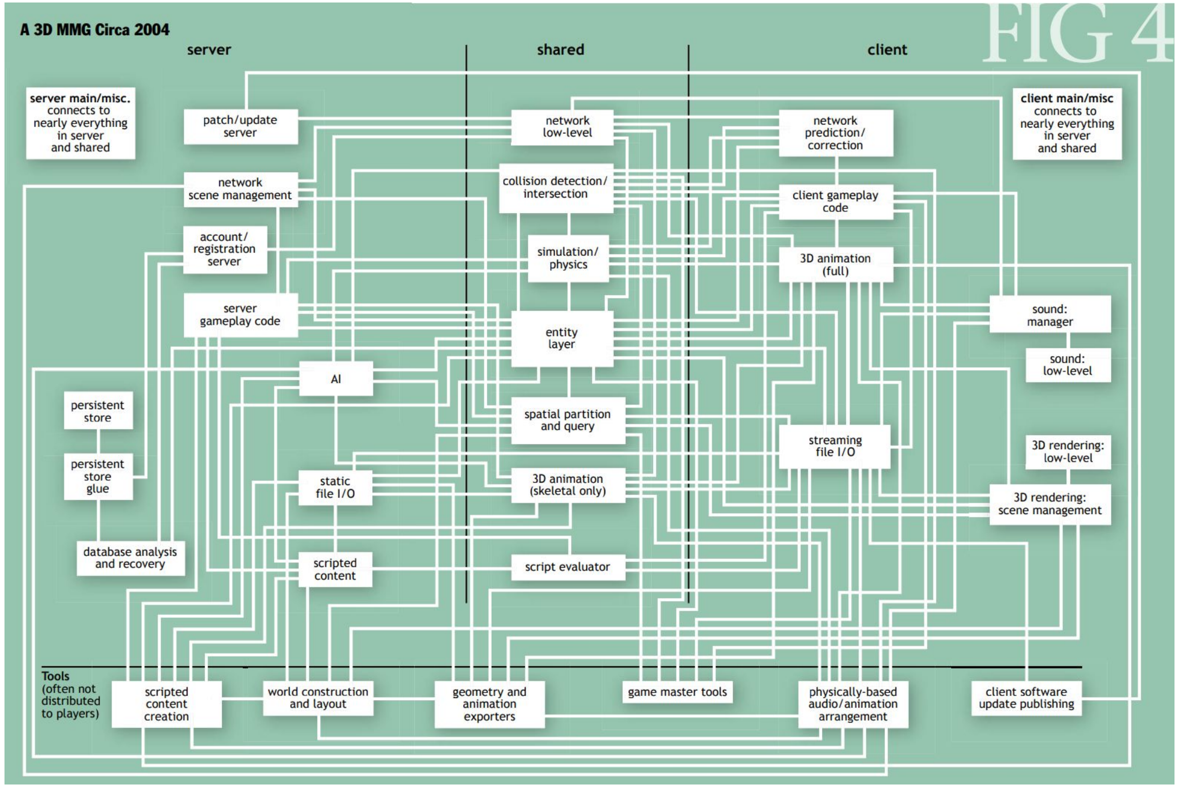
Difficulty in game development

- Past times: producing code that runs quickly
- Now(2004): getting code to produce an end result that bears some semblance to the desired functionality.
  - choosing the right high-level algorithm

- 2 main problems:
  - Problems due to highly domain-specific requirements
  - Problems due to overall project size/complexity

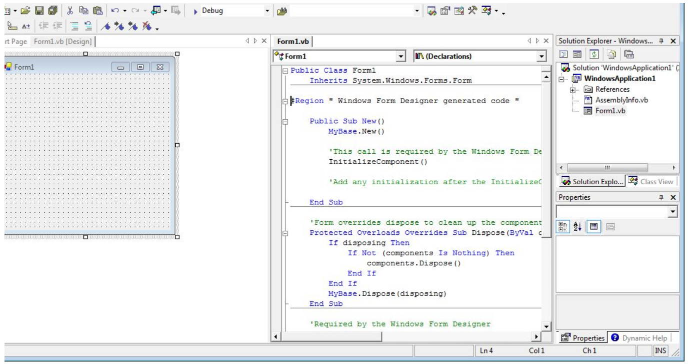
Problems due to overall project size/complexity




A 3D MMG Circa 2004
server
shared
client
FIG 4
server main/misc. connects to nearly everything in server and shared
patch/update server
network scene management
account/registration server
server gameplay code
AI
persistent store
persistent store glue
database analysis and recovery
static file I/O
scripted content
network low-level
collision detection/intersection
simulation/physics
entity layer
spatial partition and query
3D animation (skeletal only)
script evaluator
network prediction/correction
client gameplay code
3D animation (full)
streaming file I/O
client main/misc connects to nearly everything in server and shared
sound: manager
sound: low-level
3D rendering: low-level
3D rendering: scene management
Tools (often not distributed to players)
scripted content creation
world construction and layout
geometry and animation exporters
game master tools
physically-based audio/animation arrangement
client software update publishing

Tools

- Microsoft Visual Studio
  - Optimized for visual studio and c# applications
  - Useful for applications that has :heavy use of COM objects or many windows with variegated UI elements
  - Need to augment the content packages with our own plugins and other tools
- What developers actually need
  - make the system compile programs quickly
  - generate efficient code
  - produce reasonable error messages for code that uses C++ templates.
  - a fast and robust system for the games assets, 3d models, sound effects, etc



```
Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form De
        InitializeComponent()

        'Add any initialization after the InitializeC

    End Sub

    'Form overrides dispose to clean up the component
    Protected Overloads Overrides Sub Dispose(ByVal c
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
```

Workflow

- Compile, edit and debug
- Build time increases because of dependencies
    - Causes a team to put a lot of work into refactoring their code.
    - Can result in reconstruction of the whole project
    - Solution: architect the entire code base to minimize dependencies.
- Take up to half an hour to compile
- Some solutions
    - “edit and continue” feature with Visual C++
    - third-party tools to distribute compilations across many machines.
- Unavoidable start-up time

Multiplatform Development

- Build the game for all build types (debug, release) for all targeted platforms(PC,Playstation,Xbox)
- This can cause compile-time or runtime error, disrupting the work of the rest of the programming team
- In order to avoid these problems the programmer has to recompile between two or five times.
- Build Masters
- The results is that a developer has a lot of barriers

Leveraging third-party products

- Audio low-level
- Rendering low-level
- Scene management
- Collision detection and physics
- Networking low-level
- Skeletal animation and morph targets
- Persistent object storage
- Scripting languages

- Difficulty integrating these modules
  - May require domain-specific knowledge
  - Wrapper layers
  - May fail when problem it solves is smaller than the amount of work the team has to do in order to implement it
- Cost/benefit analysis

Licencing an entire engine

- Commercial engines
  - Unity
  - Unreal
  - CryEngine
- Proprietary engines
  - Frostbite
  - id Tech

- Pricing concerns
  - Article say this is expensive, but it's not really the case anymore
  - For example
    - Unity: Price per month per seat ($125 * num people * num months)
    - Unreal: 5% royalty on gross product revenue after the first $3,000
    - So let's say you make $1m with 5 people over 1 year, that's $7,500 for Unity and $50,000 for Unreal

Highly domain-specific requirements

- Script code vs. gameplay code vs. engine code
- Engine code
    - High requirements for performance and quality
    - Mathematical knowledge
        - Linear algebra
        - Rendering / graphics
        - Physics simulation
    - Algorithmic knowledge
        - Spatial partitioning, intersection and clipping of geometric primitives etc.
        - Design patterns
    - Realtime concerns

Further problems and concerns

- Depth of simulation
  - Integrating quantities over time using numerical methods
  - Skipping events (tunneling) when simulation tics are too long
  - n^2 problem => culling
- Profiling
  - "Unfortunately, there are no good profilers for games" => not really true anymore
- Increased technical complexity => Increased risk

Conclusion

Games are hard