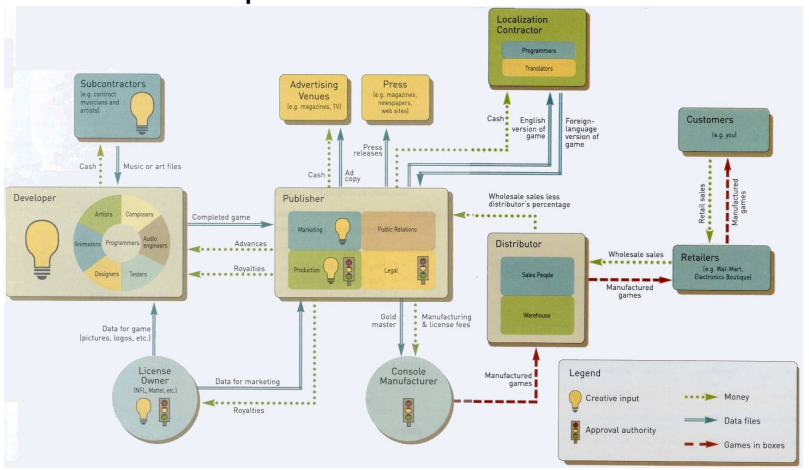
3. Game Development

3.1. Development and Publication



Example: Planning a Game Project

- Programming
  - Game Core
  - 3D Engine
  - 2D Menu and Maps
  - Physics, Collision Detection, AI, Spatial Partitioning and Special Effects
  - Sound
  - Scripted Content

2-5 persons for 2 years
⇒ 4-10 MY

- Graphics & Design
  - Look & Feel, Game Atmosphere
  - Level Design (architecture), Interior Design (objects)
  - Characters: Design and 3D Model
  - Animations
  - Render Sequences (e.g. for intro or between levels)
  - Menu and Maps (design and logic)

3-6 persons for 2 years
⇒ 6-12 MY

Example: Planning a Game Project (cont'd)

- Sound Design
  - Object Sound
  - Background Music
  - Sound Sources and Areas
  - Spoken Text Sequences

1-2 persons for 1 year
⇒ 1-2 MY

- Writing
  - Story Line
  - Documentation
  - Manual
  - In-Game Text
  - Localization

1 person for 1-2 year
⇒ 1-2 MY

Example: Planning a Game Project (cont'd)

- Networking
  - Network Interface and Protocol
  - Client-Server Programming

1 person for 2 years
⇒ 2 MY

- Tools Development
  - Physically-based Audio/Animation Arrangements
  - Scripted Content / Event Creation
  - Geometry and Animation Exporters
  - Game Master Tools

2 persons for 2 years
⇒ 4 MY

- Testing
  - Technical Design Rules
  - Feedback, Corrections, Bug Fixes

1 person for 2 years
⇒ 2 MY

Example: Planning a Game Project (cont'd)

- Project Management
  - Business Plan, Project Plan
  - Marketing, Promotion, Distribution, Sales, Finance
  - Project Manager, Project Leader
  - Technical Director
  - Director of Arts and Design
  - Location Manager for Tools/Software/Hardware
  - Human Resource Management

1-2 persons for 3 years
⇒ 3-6 MY

Total Budget: 22-32 MY

26 MY at NZ$ 50000 / year ⇒ 1.3 million NZ$

3.2. Game Design Planning

- What makes a game successful?
  - You tell me! ☺
- Game Design planning
  - Basic idea, genre, features
  - Market research
    - Who are the target groups?
  - Initial sketches (characters, scenes, …)
  - Consultations between Artists and Programmers

Game Psychology

- Identification
  - Functional roles (FRPG)
  - Group formation (collaboration)
  - Immersion
- Plans and themes
  - Averting thread, defense, attack, tasks
- Curiosity
- Challenge
- Creativity
- Success / Rewards

Take into account cultural difference!

Design Document

1. Game description
2. Definition of target groups and market research
3. System requirements (minimal recommended)
4. Characters
   - 4.1 Groups
   - 4.2 Character descriptions
   - 4.3 Motion description of main characters
5. Game relevant components
   - 5.1 Weapons
   - 5.2 Buildings
   - 5.3 Special objects
   - 5.4 Enemies

Design Document (cont'd)

6. Levels
   - 6.1 Overview
   - 6.2 Enemies
   - 6.3 Equipment / objects
   - 6.4 Special Hints
7. Scripted Events
   - 7.1 In-game events
   - 7.2 Scripted sequences between levels
8. Sound and Effects
   - 8.1 Background music
   - 8.2 Sound effects
   - 8.3 Graphic effects

Design Document (cont'd)

9. Multiplayer Mode
   - 9.1 Networking
   - 9.2 Game modes
   - 9.3 Levels
10. Interface
    - 10.1 Game menus
    - 10.2 In-game menus
    - 10.3 Controls
11. Documentation

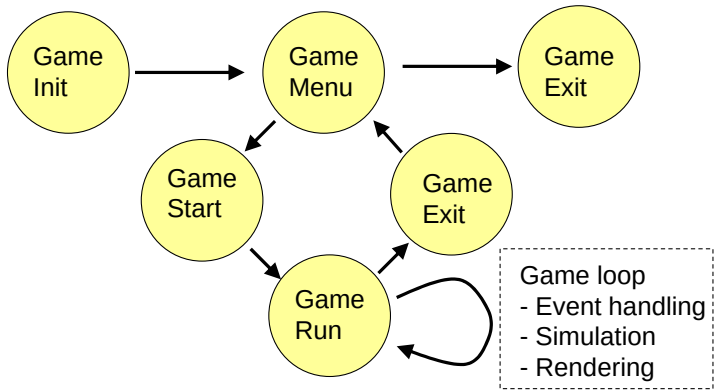
3.3. Game Level Design

- Important issues
  - Interaction between game engine and levels
  - Easy start, help, hints
  - Creative levels (new solutions)
  - Level strategy and tactic
  - Strategic and alternative goals
  - Short maneuvers, long fights
  - Rewards
  - Realism
  - Difficulty
  - Testing

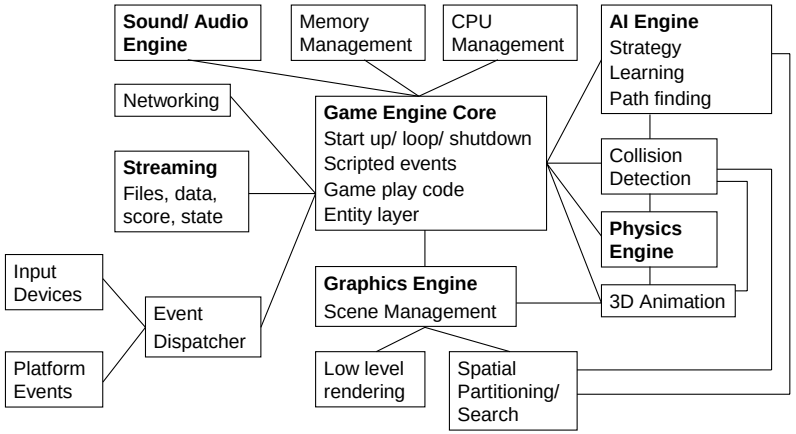
Level Design Document

- General Description
  - Title
  - Short description
  - Overview
- Detailed Description
  - Level theme (story and strategy)
  - Background story
  - Strategic goals
  - Terrain maps, Region maps
  - Starting positions
  - Distribution and map of resources
  - Game actions / event handling
  - Level specific descriptions
    - Objects, geometry, textures, light, Artifical Intelligence

3.4. Event Flow and Modules of a Game



Game Engine and Modules

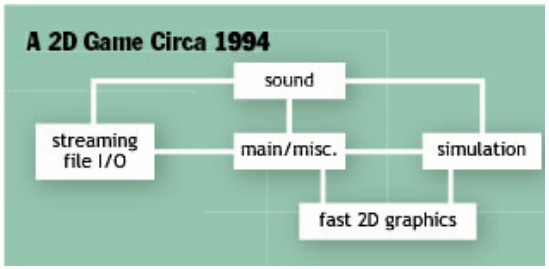


3.5. Challenges in Game Development

- Project Size and Complexity
- Tools
- Workflow and Multiplatform Development
- Third Party Components / Full Figure Option
- Highly Domain-Specific Requirements
- Depth of Simulation
- Profiling and Risk Management

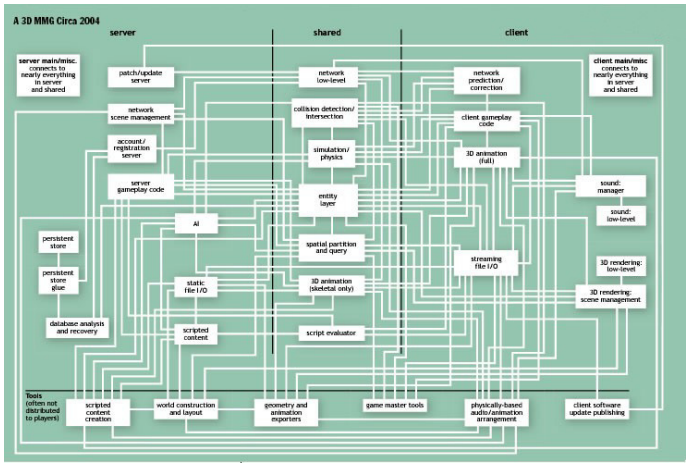
Project Size and Complexity

- The complexity of computer games has increased exponentially



© Jonathan Blow, Game Development, Vol. 1, No. 10, February 2004

Project Size and Complexity (cont'd)



Tools

Because of the complexity of game projects excellent development tools are essential …. but:

- No suitable IDE available
  - Visual C++ is the best one available
  - But too much emphasis on COM objects and UI elements
  - Compiler too slow
  - Generated code not efficient
  - No appropriate error messages for code containing C++ templates
- Very few development tools for consoles available since console life cycle too short.
- Modelling and Animation tools not suitable for real-time animations

Workflow and Multiplatform Management

- Compile/Edit/Debug cycle must be kept short
  - Refactoring
  - Distribute compiles across multiple machines (e.g. *Incredibuild*)
- Minimise start-up times for testing
- Difficult to set up proper conditions to exercise a code path
- Similar problems for content development (e.g. change of texture and model)
  - Can be handled by adding appropriate tools to the game engine
- Must maintain consistency of code during multiplatform development

Third Party Components / Full Figure Option

- Develop all components from scratch
- Use third-party components
  - Low-level audio
  - Low-level rendering
  - Scene management
  - Collision detection and physics
  - Skeletal animation and morph targets
  - No useful product for AI functionality exists!
- Use a third-party game engine
  - Quake, Unreal (professional)
  - Fly 3D, The Nebula Device, Torque (Shareware/Public Domain)
    - List of 653 (!) game engines: http://cg.cs.tu-berlin.de/~ki/engines.html

Third Party Components / Full Figure Option (cont'd)

- Disadvantages of using third-party components
  - Difficult to integrate
  - Often require "thick" interface layers
  - Often not properly tested
  - Often CPU intensive ⇒ performance bottleneck
  - Licensing costs
- Disadvantages of third-party game engines
  - Costs!! (300000-600000 US$ licensing cost for a professional engine)
  - You may not be allowed to keep copyright (GNU general public license)
  - Potentially poor fit
  - Potentially unreliable

Highly Domain-Specific Requirements

- Three level of programming in games
  - Script code
  - Gameplay code
  - Engine code (the most difficult part)
- Engine code requires
  - Advanced mathematical knowledge
  - Algorithmic knowledge
  - AI, Physics, Networking, ….
- Crosscutting concerns
  - Complex algorithms coupled together in a tight systems may clash
  - Difficult to fulfill multiple constraints

Depth of Simulation

- Games try to simulate the real world (immersive virtual environments)
  - Initial focus was on graphics (realistic rendering)
  - Now physics and AI increasingly important
- Problems
  - Complex systems are in general represented by numerical equations
  - Must avoid discontinuous functions
  - World events must be specified at a high level
  - Difficult to integrate "stiff" equations
  - Problem of tunneling

Profiling and Risk Management

- Must profile rendering and simulation performance
  - Make full use of CPU/GPU
  - Identify bottlenecks
- Manage technical risks
  - Game development must be able to proceed regardless of failure/delay
  - Carefully plan development budget
- Manage game design risk
  - Make user studies

3.6. Modelling and Animation Software

The most popular software packages are

- **Maya** (Alias)
- **3D Studio Max** (Discreet)
- **Houdini** (Side Effects Software)
- **LightWave 3D** (NewTek)
- **Softimage | XSI** (Softimage)

Example – 3D Studio Max

- Models created by starting with a spline surfaces or by transforming simple geometric objects (e.g. cubes). The resulting meshes can be edited, refined, smoothed and combined.