Developing games with COTS

A solution to escalating costs

and expanding deadlines?

Eelke Folmer
Assistant Professor CS&E
University of Nevada, Reno
http://research.eelke.com
Games have evolved

Super Mario (1987)
Problems with game dev

Technological advances & games showcasing these advances continuously push the boundaries what is to be expected of games.

Resources required to produce games have significantly increased:

- 1992: $350,000 | 12 people | 6-12 months
- 2005: $3M - $10M | 25 people | 18 -24 months

Price of computer games has remained the same.

Conclusion: you have to sell a lot of games to make a profit.

AAA titles > 500,000 copies
Dilemma

In order to survive game developers must find a way to:

★ Sell more games
  – Hits driven: top 99 games (only 3.3% of developed games) accounts for 55% of all sales.
  – Only 1 in 7 games makes a profit.

★ Find a way to lower costs & development time
  – Reuse? COTS?

Benefits:
★ Lower cost & development time
★ Higher quality of COTS & game
★ Advance technology at a faster rate.
Use of COTS in games not new:
★ Game engines (ID: doom / unreal) have been around for a decade.
★ Recently: lots of COTS entering the market specialized in less well understood game areas (physics / AI).
Reference Architecture

Provide a RA to:

- Discuss commonality between games.
- Sketch out areas of reuse.

Game interface
- Gui framework
- Graphics
- Sound
- AI
- Physics
- Network

Domain specific

Hardware abstraction
- Input
- Graphics
- Audio
- Network

Platform software
- Wii/Xbox/playstation

Specific

General
Six areas of reuse

- Network ~ server - client communication
- Graphics
  - Rendering ~ pixelpushing
  - Modeling ~ managing game objects
  - Animation ~ creating realistic movement
  - Texturing & effects ~ bump mapping
- Gui ~ building interfaces
- AI ~ creating the illusion of intelligence
- Physics ~ adhere to newton’s law
- Sound ~ music/sound

Not part of the game but of the “content pipeline”

TOOLS
Future of COTS?

Four problem areas worth further investigating:
1. COTS vs framework
2. Complexity & SA design
3. The “emerging” architecture
4. Buy or build?

more cots == good
COTS vs Frameworks

# available COTS increases -> integration becomes a problem.

Will game COTS end up like J2EE or .NET?

- integration problems
- More control

- no integration problems
- Less flexibility

cots

frameworks
Complexity & SA design

Complexity mainly due to increase in #components, but also a spaghetti of dependencies making your game less flexible & expandable.
Causes of complexity

Internal data representation

Graphics  Sound  Physics

update  update  change

Object centric view

Game

Sound  Physics

Graphics

Better???
The “emerging” architecture

- Ad hoc design is commonplace.
- An architecture “emerges”
- Architecture may not be optimal for your game

- Connectors play a fundamental role in Achieving Quality
- Needs to be further explored in the domain of games.
How do you know the COTS provides what you need?
★ Requires deep knowledge of COTS.
★ Game development is explorative with frequently changing requirements.
★ Avoid end up rewriting most of the COTS’ functionality.

★ Guidelines for component selection?
★ Increase flexibility while still meeting perf. Req.?
Questions?