Virtual Worlds
Lessons from the Bleeding Edge of Multiplayer Gaming

Greg Corson
Dave McCoy

State of the Community
1990

- Artists often used as “subcontractors”
- Many dominant 3D tools are SGI based and very expensive
- 2D and 2-1/2 games dominant
- A few games dabbling in full 3D
- A lot of 3D graphic content for games is generated with graph paper
Inspiration

- Academia
- Film and Video
- Arcade Games
- Military Simulators

Academia

- Virtual Reality was about to take over everyone’s life
- Scientific Visualization with computer graphics
Film & Video Effects
(my background)

• TRON
• Last Starfighter
• Abel & Associates commercials
  – Sexy Robot
  – Levi’s
• PDI

Arcade Games

• Mostly sprite based
• Cinematronics vector display wire frame 3D
  – Tail Gunner
  – Red Baron
  – Battlezone
• I-Robot – Filled Polygons
• W Industries Virtuality – Dactyl Nightmare
Military Simulators

• Evans & Sutherland
• GE
• Hughes/Loral – Simnet
• Multigen tools

Problem 1 – Setting the Bar

What can we do?
What can we do?

- As game technology capabilities accelerate this question becomes ever more important
- Greg and I have both been in support roles for new graphics technology – this is the most common question
- Then and now the answer from experts can never be complete

What can we do?

- $25,000 proprietary graphic system
- Graphics board is $8,000 with volume discount
- Hard to find experts in the field in art or code – most real-time graphic experts in military simulation
The Up Side

No established limitations

Benchmark

• Never trust the manufacturers numbers
  – (or delivery dates)
Benchmark

• Test what the system characteristics and performance are under game circumstances
  Display resolutions
  Polygon size
  Position and animation updates
  Hierarchies and state changes
  Lighting

• Display characteristics
• Geometry performance
• Fill-rate performance

System 2 Characteristics

• 600-700 polygons per frame @ 12 Hz Floating
• 640 x 480 24 bit display
• No Z-Buffer – draw order, BSP and centerpoint sorting graphic language
• Flat shading
• Background image
• RGBI color definition
• Bitmap transparency
• 16 light sources
• Per vertex fog
Tesla Characteristics

- 1500 polygons per frame @ 24 Hz
- Programmable Hardware
- Multiple primitive types
- 800 x 600 24 bit display
- Z-buffer tiles
- 768 K texture
- Multiple texture modes

Tesla Characteristics

- 2 light sources
- Screen door and single bit transparency
- Per vertex color & transparency
- Per pixel fog – range based
- Fire and forget rendering
- Low bandwidth pipe
“Nothing so focuses a man’s mind as the knowledge that he will be hanged in the morning”

Problem 2
What Should We Do?
Limitations focused our efforts
Play to your audience

Exploit your unique strengths

Beauty in Context

- Design all art in context of game usage
- Evaluate all art under actual gaming conditions
- Focus time on most common game conditions – similar to programming
- Don’t let very unlikely or very uncommon conditions prevent the use of techniques
Beauty in Context

• Be consistent in visual communication
• Use visual touchstones
• Avoid monoscale
• Imply detail beyond perception
• LOD on more than just distance
• Don’t just LOD geometry

It’s not the actual image that matters
People see what they think they see
CHEAT! CHEAT! CHEAT!

Rely on non-graphic resources for graphic solutions
Game Design

- Limit viewpoints
- Distribute visual complexity
- Channel attention
- Consider when people will have time to focus and when will they be harried

Game Design

- Game design and art must be thematically consistent to reinforce both
- What are you both trying to say?
- Avoid errors of diversity over quality
- How should the player feel?
Audio

- Film knows its power – watch dailies
- With early systems we had no choice but to rely on audio for 50% of the image
- Supposedly everyone knows this now, but do they really practice it?
- Audio is now where art once was

Audio

- For Tesla, repeated concessions on visuals were made for top quality audio
- It worked
Audio

• Sound and music generally follow – sometimes they should lead

System 2 Strengths

• Background image
• Fog
• Graphic Language
• Lighting
• RGBI
Tesla Strengths

• Optics
• Programmable Hardware
• Texture Modes
• Fire & Forget Features
• Hardware Particle Language
• Primitives (spheres)
• Fill rate

Problem 3 – How do we get it done?

(in time)
System 2 and Tesla both simultaneous code/art development

- As with many current projects, a production necessity - but not the way to work
- More tenable with small teams
- Art and code have different development cycles
- Leads to significant waste

Common Project Progression

- Concept
- Design prototyping = design doc outline
- Art prototyping = concept art
- Concept art and design doc used to sell project
- Project sells
- Programming prototyping begins
- Art and design production begin
System 2 Production Begins

- Functional Game Design complete
- 2 Programmers – 1 Artist (game, UI, publicity)
- Uncompleted hardware
- No code base
- No tools
- Legacy graphic language (BSP)

Art Production
(Phase 1 – The Beginning)

- ASCII Script Graphic Language
  - Error in scripting can crash whole system
  - Graphic Language Scripts written by artist
    (art code is worse than programmer art)
- Graph paper & coordinates visual geometry creation
- Graph paper & coordinates collision geometry creation
- Graph paper & coordinates map generation
- Manual sorting of objects
- PC Based animation system
Art Production (Phase 2 – After the Begging)

- Artist written CAD geometry conversion
- Tool to assist object creation - “Sort Of”
- Real-time animation system

Art Production (Phase 3 – After the Beatings)

- Automatic constant priority shape generator
- Object collision shape tool
- Some error checking of artist scripts
- Interactive CAD Map Generation
Art Production

• Code & Content completely intermingled

Learned & Resolved

Firewall between code & content
Gallery / Studio Model
Tesla Production Begins

- Legacy game design (major revision)
- 4 Programmers – 4 Artists
- Programmable hardware/New Architecture
- New Code

- Some tools – additional tools programming scheduled
Art Production
(Phase 1 – The Beginning)

• Separation of art & code
• Art production is largely off-the-shelf tools with conversion utilities
• Script based model files with references to other file types (collision, visual, etc.)
• Perfly model viewer
• Additional Tools Scheduled

Art Production
(Phase 2)

• Additional art production tools scheduled
Art Production (Phase 3)

• Art tools production month scheduled

Art Production (Phase 4)

• Art tools production month delayed
Art Production
(Phase 4 – After the Beating)

• Permanent assignment of art tools programmer
• Polisher – LOD texture tool
• MAP – Conversion utility for world building
• Animation creation/blending tool
• Hierarchy construction tool
• Radar/interface tool
• Artists can add and revise content

Learned & Resolved

• The fidelity problem
  Better = More
• File interdependence is a form of coding
  Confusing and dangerous
• Artist independence critical
  Production and Morale
• Iteration speed = quality
  Ability to polish
  Willingness to experiment
Learned & Resolved

• Artists determine needed art tools
• Tools maintenance is critical
• Dogfood approach
• Tools suite – not omni tools
  Too hard to extend
  Too hard to fix
  Too long in development
  Too hard to predict production issues

Studio/Gallery Model

Trying to reconcile the concerns of programmers and artists
Studio/Gallery Model

• Studio controlled by artists
• Studio can be messy & unorganized
• Studio should be amenable to experimentation
• Studio needs resources
• The studio is open even when the gallery is closed
• Studio is not the gallery

Studio/Gallery Model

• Programmers control gallery
• Admittance to gallery is by programmer invitation only
• Gallery must conform to strict rules
Studio/Gallery Model Problems

- Studio must be able to create anything gallery can display
- Studio disorganization can be a real problem for large projects
- Moving from studio to gallery can be cumbersome
- Sometimes programmers & artists argue about what should be in gallery

A technological medium

Requires code and art
Comments to Programmers

• Artists actually know some things you don’t
• Paintings are not judged by the quality of the paint

Comments to Artists

• Programming requires extreme creativity – if suggestions are offered, listen. If they aren’t, ask
• Doing is much harder than dreaming
Problem 4 – Learning for the Next One

Assuming there is a next one

Remember

• The impact of demos
• The unpredictable
• The impact of delay on the product’s quality
• The time required to revise based on testing
• The impact of assigning personnel prematurely
• The man/month fallacy
• The absolute statements that were wrong
Games
Unused Assets can equal 80%

Movies
Unused Assets can equal 95%