Networked virtual environments

SGN-5406 Virtual Reality 2012
Atanas Boev

based on material by
Stanislav Stankovic and Ismo Rakkolainen
Outline

Net-VE overview

Architectures

Shared state

Consistency

Shared repository

Frequent state regeneration

Dead reckoning

Throughput

Web3D standards

VRML

X3D

MPEG-4

JAVA-3D

WebGL

The future?
Net-VE
NETWORKED VIRTUAL ENVIRONMENTS
Networked Virtual Environments

- **Net-VE** – multiuser networked virtual environments
- **Net-VEs provide:**
  - Common interactive environment
  - Common state of environment
  - Common time
  - Means of communication
- **Avatar** – virtual user representation in the VE
  - In Hinduism – “descent of God in human and other forms”
- **Known as**
  - Net-VE, Shared VE, MUVE, DIVE, etc.
  - “Cyberspace”
NetVE applications

- VE over LAN or internet
- Military simulations
- Teleconferencing
- Distributed CAD
- Distance learning
- Online communities
  - Surged in popularity in 2000s, now in decline
- Gaming
  - Top application of Net-VE
Net-VE challenges

- **Net-VE components**
  - 3D graphics, displays
  - Processing
  - Interaction, I/O-devices
  - Networking (LAN, Internet)

- **Net-VE bottlenecks**
  - Client computer performance
  - World modeling
  - Network bandwidth
  - Server performance
NetVE examples

- **Second life**
  - Shared environment, virtual estate trade, virtual goods trade
- **World of Warcraft**
  - MMORPG – massively-multiplayer online
- **XBOX Live**
  - Many multi-player online games
- **Habbo Hotel**
  - Non-immersive NetVE
NetVE examples

Doom

http://youtu.be/53tA3yRnrPo
NetVE examples

Second Life

https://www.youtube.com/watch?v=CaLKFeJLnql
NetVE examples

Eve Online
http://youtu.be/Qa4HsN6YTIU

Skyrim
http://youtu.be/RG4S-SCWfyM
Distributed environment

• **Problems**
  • Shared world - How to organize the transmission of updates?
  • Simultaneously over the network - How to update changes to all clients?

• **Practical issues**
  • Compatibility
  • Latency
  • Speed of the network
  • Consistency
  • Heterogeneity
  • Failure management
Distributed environment example

Jam with Chrome – online music band

http://youtu.be/YkvKICWaRT4
Architectures

OVERVIEW
Net-VE Basic Architectures

1. Serverless systems (Peer to Peer)
2. Centralized client-server systems
3. Multiple server systems
4. Coordinated multiple servers

• Aims (to avoid bottlenecks)
  • Better communication models – reduce the number of connections and messages
  • Better database models – distributed databases
  • Better decision making – make it distributed, but any given decision is made in only one place
Architectures
SERVERLESS
Serverless systems - overview

- **Peer-to-peer clients on a LAN**
  - Each client broadcasts its state directly to others
  - Suitable for small networks

- **Peer-to-peer clients on a WAN**
  - Avoid sending the same message to all participants
  - Broadcast is wasteful, multicast is selective
  - Area-of-interest management (AOIM) - Assigns packets for multicast groups
Serverless systems – pros/Cons

**Pros**
- No central bottleneck, single point of failure
- Multicast is network-efficient
- Multicast subscription = filtering

**Cons**
- Difficult to manage
- Network bandwidth is a bottleneck: $O(N^2)$
- All broadcast packets must be examined by each client
Architectures

CENTRALIZED SYSTEMS
Centralized systems

- Each client communicates with the server only
- The server is distributing the state to all players
  - Limited number of players
  - Latency
Centralized systems – Pros/Cons

- **Pros**
  - Simple
  - Server can filter packets (observe fairness)

- **Cons**
  - Server is a bottleneck
  - Server reliability is a bottleneck
  - Latency
  - In order to get consistency – delay in one client delays all others
Architectures

COORDINATED MULTI-SERVER
Coordinated multiple servers

- Server hierarchy – many servers talk to each other
  - Function based
  - Location based
Coordinated servers – Pros/Cons

- **Pros**
  - Filtering
  - Dynamic load sharing
  - Can share a single world

- **Cons**
  - Coordination is difficult
  - Can increase latency
Web-based

- **High-level approach**
  - Completely separated from network topology

- **Using web technologies**
  - Clients – browsers
  - Protocol – HTTP
  - Languages – Javascript, Flash, Java, NaCl
Choice

- Depends on priorities
  - Scalability – serverless / non-coordinated
  - Reliability – coordinated
  - Simplicity – centralised
  - Interactivity – serverless

- Scalability / reusability of code – web based
Dynamic Shared State

OVERVIEW
The problem

- All participants want accurate, real-time view
  - Location and orientation of objects
  - How and when to interact
  - Environmental info (weather, terrain, …)
- However, clients are on remote computers
  - All packets have delay (latency)
  - Different latency for each
  - Limited network bandwidth
  - Packets may get lost
Trade-off: consistency vs. speed

1. Shared repository
   - Easy, consistent
   - Slow, unpredictable, overhead

2. Frequent state regeneration
   - Continuous blind broadcast
   - Easy to implement
   - Network traffic

3. Dead reckoning
   - Prediction
   - Convergence
Trade-off: consistency vs. speed

- **Consistency**
  - Consistent view at all sites,
  - Less frequent state updates

- **Throughput**
  - Each site has different view,
  - More frequent state updates

- **Shared Repository**
- **Frequent State Regeneration**
- **Dead Reckoning**
Shared State

SHARED REPOSITORY
Shared repositories

- **Common data storage**
  - Client sends updated to the storage
  - Reads from the storage

- **Suitable for**
  - Small-scale LAN systems
  - Consistency-critical systems
Shared repositories – Pros/Cons

• **Pros**
  • Easy programming model
  • Absolute state consistency

• **Cons**
  • Single point of failure
  • Bottleneck
  • Unpredictable performance
  • Communications overhead
Shared State

FREQUENT STATE REGENERATION
Frequent state update

- **Send a constant stream of state updates**
  - Typically blind network broadcasts
  - Multicasting and filtering reduces bandwidth
  - Hook into event loop or on timer
- **Frequent updates: fast recovery**
- **Common for mid-scale LAN systems**
  - E.g. Doom
Frequent state update – Pros/Cons

- **Pros**
  - Simple to implement
  - No servers needed
  - Better update throughput

- **Cons**
  - Considerable bandwidth
  - Network latency and jitter
  - Different update rates
  - No absolute consistency
Shared State

DEAD RECKONING
Dead reckoning

- Position of object is interpolated by the client
  - Using last known velocity and position
  - No need for frequent updates
- Prediction from periodic updates
  - Linear, quadratic, spline interpolation
  - What speed is the moving object capable of?
- Convergence
  - Fusion of predicted with real data (once real data are received
    - Snap, linear, quadratic, spline, etc
- Best for large-scale WAN systems permitting inexact state consistency
Dead reckoning

1. Update
   - Actual position
   - Actual velocity

2. Prediction
   - Predicted position
   - Predicted velocity

3. Convergence
   - Actual position
   - Converged position
   - Predicted position
Dear reckoning – Pros/Cons

**Pros**
- Insensitive to network latency
- Low-frequency updates, reduced bandwidth

**Cons**
- Does not guarantee identical states
- More complex algorithms
- Prediction model is object-specific
- Prediction errors significant over poor networks
Case study: Lissu Liikenteenseuranta

- Real-time information for Tampere busses
  - Knows position and movement direction
  - Predicts position based on last information
  - Precision is not critical

http://lissu.tampere.fi/
Web3D

OVERVIEW
What is Web3D

- **3D extension for the Internet**
  - HTML: 2D documents
  - SVG: 2D vector graphics in XML
  - Web3D: 3D environments

- **VR for the masses**
  - Not (necessarily) visually immersive
  - Not (necessarily) on 3D displays
  - Not (necessarily) using 3D tracking

- **More compact than a video clip**
  - Relatively small files
  - View of angle, resolution is not limited

- **Also used off-line, stand-alone**
Web3D applications

- Architecture, guidance
  - Tampere 3D City Info (1998-2001)
  - Google Earth, Street View etc.
  - Google Earth (Mars, Moon, Sky),
  - MS Virtual Earth 3D (now Bing maps platform)
- Art, history
  - Many small projects
- Entertainment, games
- Science, visualization
- Product promotion, instructions
- 3D page of the week
  www.parallelgraphics.com/products/cortona/best
Web3D

VRML
History of VRML

- **VRML = Virtual reality modelling language**
- **Appeared almost simultaneously with the www**
  - WWW1-conference 1994
  - VRML 1.0 May 1995
    - Geometry description based on SGI Open Inventor
    - Based on objects + text, polygons, materials, lightning
  - VRML 2.0 August 1996 = VRML97
    - ISO standard
    - More efficient structure
    - New possibilities: Movement, interaction, collision detection, programmability
    - New objects: sensors, routes, protos, scripts, interpolators, sounds, etc.
#VRML V2.0 utf8
Transform {
  children [
    Shape {
      geometry Sphere {}
      appearance Appearance {
        material Material { diffuseColor 1 0 0 }
      }
    }
  ]
}
VRML - interactions

- **Sensors** *(SoundSensor, TouchSensor)*
  - A notification of an event (distance, time, visibility)

- **ROUTE**
  - Connects event in a node to another node
  - Describes functional behaviour *(if event then action)*

- **Script-node extends possibilities**
  - If *event* then *script*
  - ECMAScript, Java, JavaScript, ...
  - Multi-user, physical simulations, etc

```plaintext
ROUTE Node.eventOutName_changed TO Node.set_eventInName
DEF CLICKER TouchSensor { enabled TRUE }
DEF LIGHT DirectionalLight { on FALSE }
ROUTE CLICKER.enabled_changed TO LIGHT.set_on
If touch then light
```
The example plays Westminster Chimes once an hour. Uses sounder and Sensor.

```vrml
#VRML V2.0 utf8
Group { children [ 
   DEF Hour TimeSensor { 
       loop TRUE 
       cycleInterval 3600.0 # seconds in 1 hour 
   } 
   Sound { 
       source DEF Sounder AudioClip { 
           url "http://...../westminster.mid" } 
   } 
] 
ROUTE Hour.cycleTime TO Sounder.startTime
```
Interpolators, Viewpoints and Sounds

- **Interpolators**
  - change properties over time
  - changes of direction, trajectories, colors, forms

- **Viewpoints**
  - Predefined “camera” positions
  - Makes viewing easier

- **AudioClip**
  - Wave, MIDI

- **Sound**
  - Ambient
  - Changes linearly (0,1)
  - Attenuates fully after a given distance
Web3D
X3D
• **X3D = eXtendible 3D**
  • Does not compete with existing standards, but provides interoperability

• **The “modern version” of VRML**
  • Uses XML: HTML-looking
  • VRML 97: DEF MyView viewpoint \{ position 0 0 10 \}
  • XML: `<viewpoint id='MyView' position='0 0 10' >`

• **Open Standard in 2002**
  • Adopted by MPEG-4 group
  • Cooperation with W3C, Open Source, Java3D, Khronos

• **X3D Binary**
  • compressed X3D files
  • Approved by ISO/IEC in 2007

• **Availability**
  • X3D Browsers
  • Browser plugins
  • Java applets
  • Check web3d.org
X3D

- **Supports**
  - 2D, 3D, CAD, animation
  - spatialized audio and video,
  - user interaction, navigation
  - user-defined data types, scripting
  - networking
  - physical simulation

- **X3D profiles**
  - Core (lightweight, fast)
  - Extensions
  - VRML97 profile
X3D examples

http://youtu.be/f74fEynl3Gc
Web3D
MPEG-4
MPEG

- MPEG = Moving Picture Experts Group
- MPEG-1: Digital audio/video
- MPEG-2: Digital TV signal compression
- MPEG-4: Multimedia streaming content
- MPEG-7: Digital information searching
- MPEG-21: Multimedia Framework
MPEG-4

- A wide standard
  - http://www.m4if.org/mpeg4/
  - Open, but not public
- Designed for
  - Digital television
  - Interactive graphics (synthetic objects)
  - Interactive multimedia (www etc.)
- Consists of several parts
  - Systems, Visual, Audio, Conformance test, Reference software, DMIF, delivery
MPEG-4 objects

- Visual objects - Images, sprites, graphics, video
- Audio - Natural or structured audio, Text-to-Speech, talking heads
- Vector-based - 2D/3D, CG objects, VRML scenes, text, faces, avatars
- Compound objects - Groups primitives together (sub-trees)
MPEG-4 structure

- **BIFS = Binary Format for Scene**
- **Contains MPEG-4 objects**
- **Can be file or stream**
MPEG-4 problem

- “If you build it, they will NOT come”
  - Scene description standard with no scene creation tools is doomed to fail

- MPEG-4 describes complex scene representation, but does not provide tools for creating such scenes
  - E.g. convert video to 3D objects

- No one implemented the full standard
  - Most implementations support only MPEG-4 baseline standard (video stream)
  - Closer implementation – IAVAS player by TU-Ilmenau
Web3D
JAVA-3D
Java 3D

Java API for 3D

- Scene graph-based (VRML loader, etc.)
- Cross-platform, device independent
- Allows building of standalone apps or java applets

- https://java3d.dev.java.net/
- Might be replaced with JavaFX
Java 3D code example

```java
import com.sun.j3d.utils.applet.MainFrame;
import ...
public class HelloSphere extends Applet {
    public HelloSphere() {
        setLayout(new BorderLayout());
        Canvas3D canvas3D = new Canvas3D(null);
        add("Center", canvas3D);
        BranchGroup scene =
            createSceneGraph();
        scene.compile();
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewingPlatform().setNominalViewingTransform();
        simpleU.addBranchGraph(scene);
    }  // end of HelloSphere (constructor)

    public BranchGroup createSceneGraph() {
        BranchGroup objRoot = new BranchGroup();
        ColoringAttributes ca = new ColoringAttributes();
        ca.setColor(1.0f, 0.0f, 0.0f);
        Appearance sphereApp = new Appearance();
        sphereApp.setColoringAttributes(ca);
        Sphere sphere1 = new Sphere(0.4f);
        sphere1.setAppearance(sphereApp);
        objRoot.addChild(sphere1);
        return objRoot;
    }  // end of createSceneGraph method

    public static void main(String[] args) {
        Frame frame = new MainFrame(new HelloSphere(), 256, 256);
    }  // end of main (method of HelloSphere)
}  // end of class HelloSphere

→ Complex scene description, imperative coding style
```
Java 3D example

Java 3D Game Engine

http://youtu.be/YVvBEFOj_T8
Web3D
WEBGL
WebGL

- 3D Rendering API
- Based on web-technologies
  - JavaScript library for rendering of 3D objects
  - Works with HTML5 `<canvas>` tag.
  - Browser support: Firefox, Chrome 9, Safari 5.1 (Lion), Opera 12.
- Extension of OpenGL
- Maintained by Khronos Group (Apple, AMD/ATI, Interl, Nokia, Nvidia, Microsoft, etc.)
- Support for GPU acceleration
WebGL demo

http://vimeo.com/27714194
WebGL water, by @evanwallace
NetVE

THE FUTURE?
The future of Web3D

- 3D imaging is becoming popular
  - Games
  - Web pages (360 degree view of a product)
  - Stereoscopic 3D, Youtube 3D

- 3D GUI is becoming popular
  - Interaction in 3D

- Many competing technologies
  - WebGL
  - X3D on the rise
  - (lots of) proprietary engines
Sources

- web3d.org
- Publications
  - IEEE Computer Graphics & Applications
- Conferences
  - SIGGRAPH, Web3D
- Books:
  - Schreer – Kauff – Sikora: 3D Videocommunication
  - Ozaktas - Onural: Three-Dimensional Television, 2007
  - Javidi, Okano (Eds.): 3D TV, Video and Display Tech.
  - Brutzman, Daly, X3D: Extensible 3D Graphics for Web Authors, 2007
More books

Walsh - Bourges-Sévenier: Core Web3D

Polevoi:
Interactive Web Graphics with Shout3D

Hartman - Wernecke:
VRML 2.0 Handbook

Roehl et al.:
Late Night VRML 2.0 with Java

Barrilleaux: 3D User Interfaces With Java 3D

Carey - Bell:
The Annotated VRML 97 Reference
http://www.best.com/~rikk/Book/