Project Wonderland
Open-source virtual world

Jonathan Kaplan,
John Harris, Justin Rounds
Sun Microsystems Laboratories

Agenda

- **What is Wonderland?**
- Getting and running Wonderland
- Adding artwork
- Programming new features
- Conclusion

Research Motivation

- Over 50% of Sun's workforce is out of the office on any given day
- Challenges of remote work
  - Everybody is remote
  - Lack of social interaction
  - Difficulty brainstorming
- Could virtual worlds be the solution?

Platform Motivation

- Researched a number of other platforms
  - Good news: virtual worlds could solve the problems
  - Bad news: none of the existing toolkits met our needs
- Is the platform:
  - Extensible?
  - Scalable?
  - Easy to program?
  - Open source?

Major Features

- Meeting Central - High-fidelity, immersive audio
- Looking Glass - Application sharing in 3D world
- Darkstar - Scalable from laptop to server cluster
- All - Java-based, highly extensible

Architecture

World

MPK20: Sun's Virtual Workplace
- World customized to support Sun's distributed workforce
- Includes applications for sharing and collaboration

Client

Project Wonderland
- Open source Java 3D-based graphics engine
- Manages world, animation, and avatars
- Supports app sharing (initially Java and X apps)
- Extensible and customizable worlds

Software Phone

Server

Project Darkstar
- Open source communication and app framework
- Targeted at games
- Highly scalable
- Handles persistence
- Allows extensible set of core services

Voice Bridge

Demo Video

Agenda

What is Wonderland?

➔ **Getting and running Wonderland**

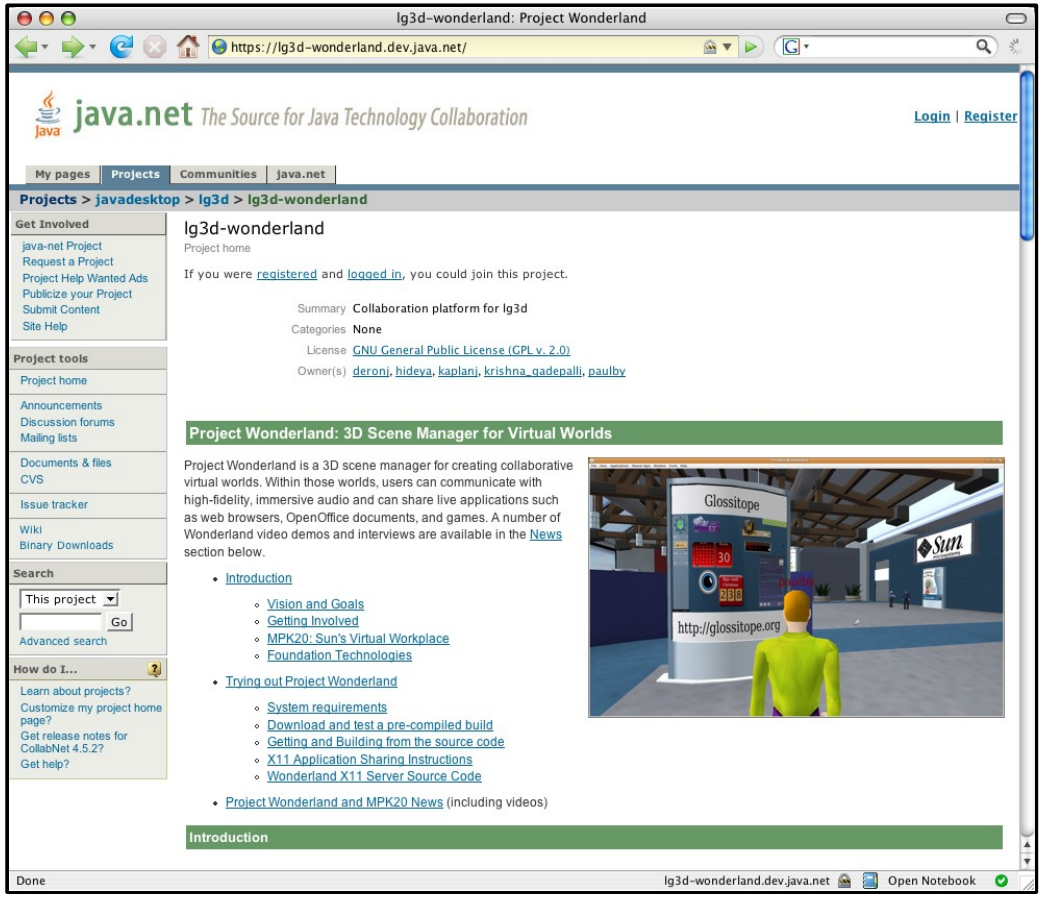
Adding artwork

Programming new features

Conclusion

Open Source Project

- http://wonderland.dev.java.net
- downloads
- source
- forums
- wikis

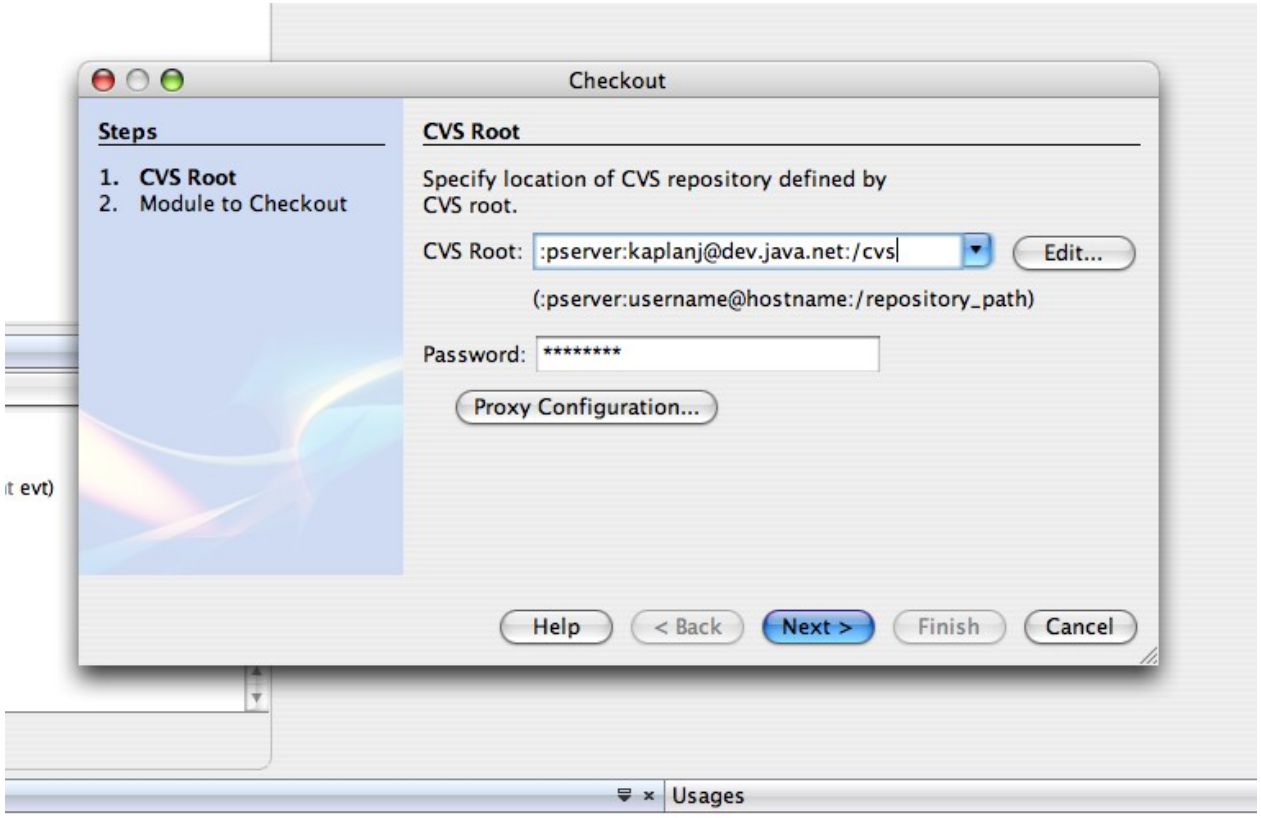


Technical Requirements

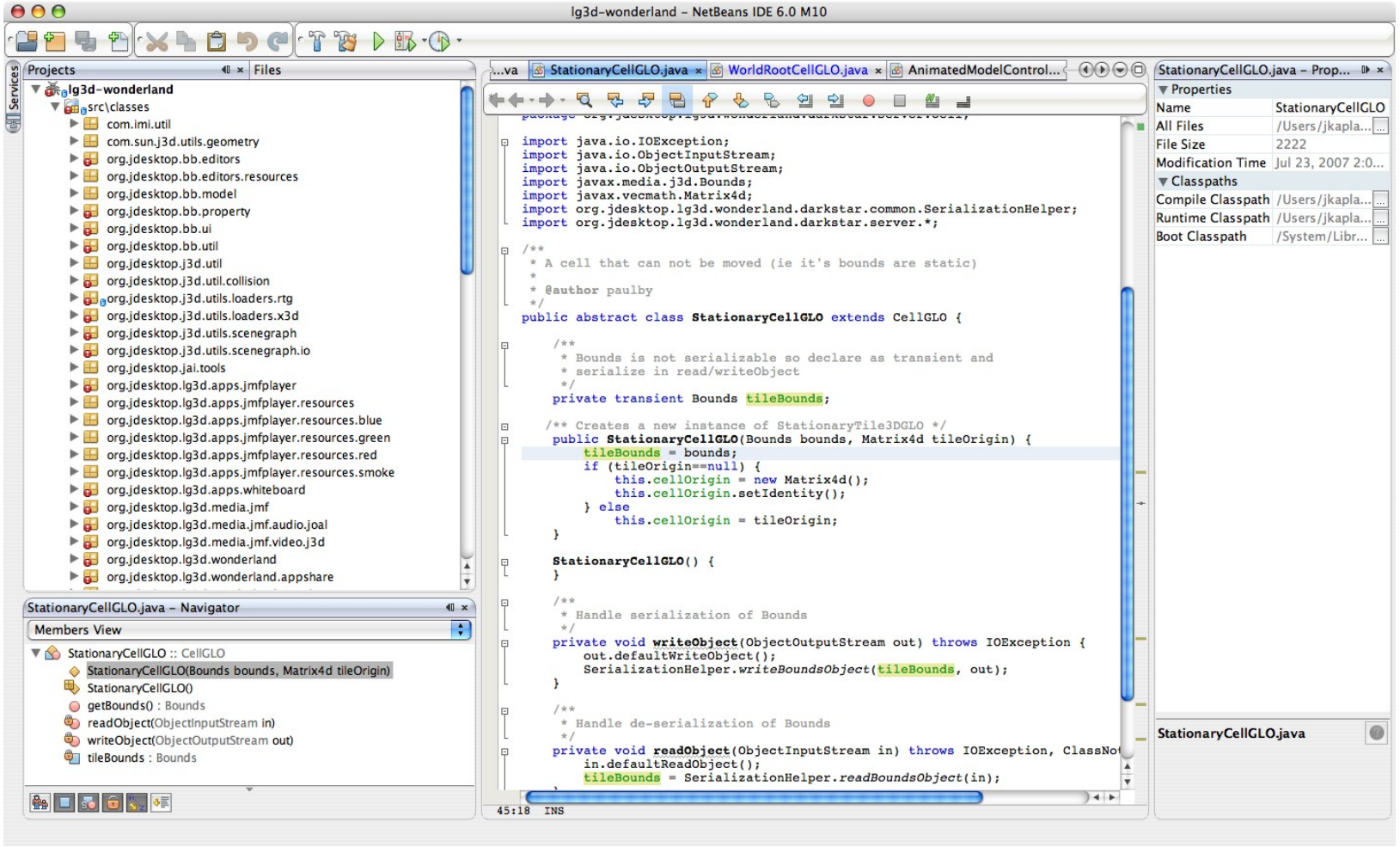
- “Modern game hardware”
- 3D accelerated graphics
  - 128MB video memory
  - ATI or nVIDIA
- ~1GB RAM
- Linux, Windows, Mac, Solaris
- Java 6

Check-out

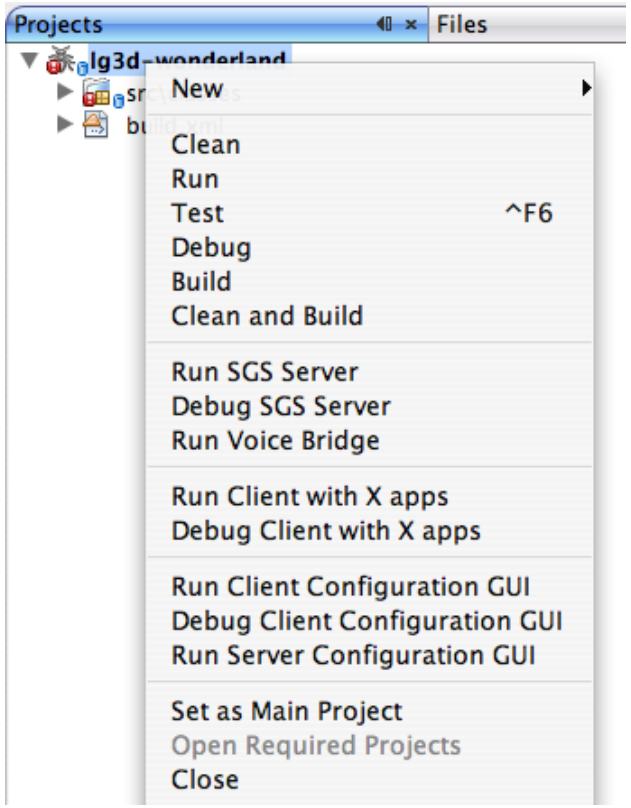
- cvs -d :pserver:*username*@dev.java.net:/cvs co lg3d-wonderland



Open in NetBeans



Run, Run, Run



1. Run voice bridge (optional)
   # ant run-bridge
2. Run Darkstar server
   # ant run-sgs
3. Run Client
   # ant run

Agenda

What is Wonderland?

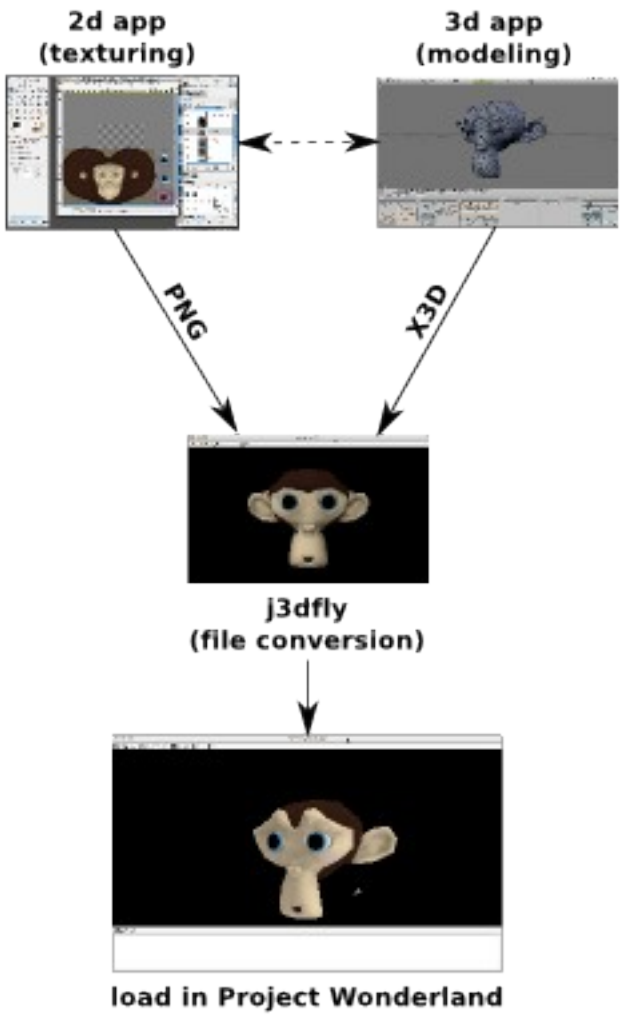
Getting and running Wonderland

➔ **Adding artwork**

Programming new features

Conclusion

Overview of Art Path



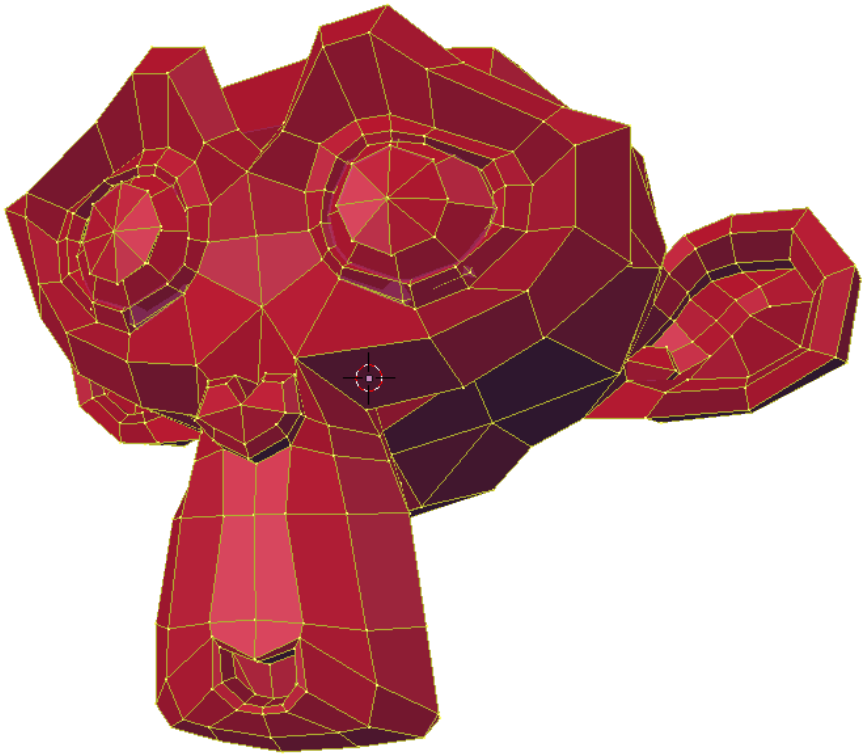
- Open art path, use existing tools
  - Commerical: Maya, Photoshop
  - Free: Blender, Gimp
- Use x3d as interchange format
  - ISO standard, defined by Web3D consortium
  - XML file format
  - Follow-on to VRML
- J3dFly translator

Setup Wonderland

- Normally artwork is downloaded from the web
  - Need to modify to use your artwork
- Artwork is open source too
  - cvs project lg3d-wonderland-art
- Edit lg3d-wonderland/my.build.properties
  - `wonderland.useLocalArt = true`

Create the Art

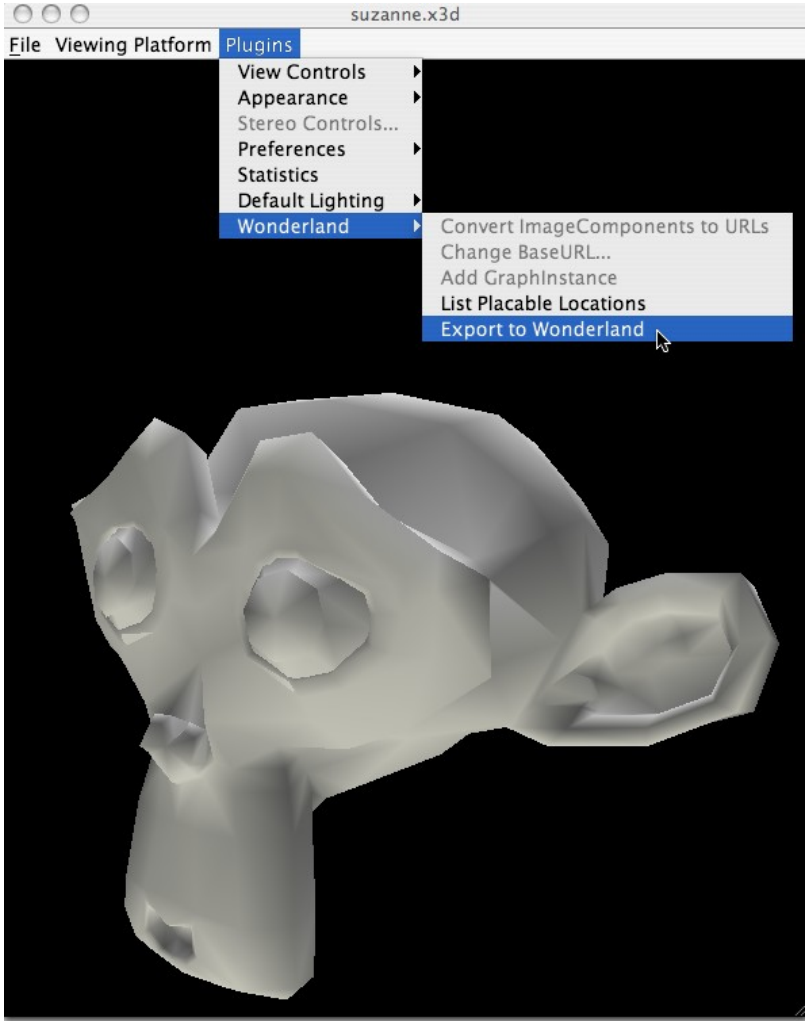
- Using Blender
- Modeling for real-time
  - Low polygon count
  - Normals are important
  - Continuous meshes!
- Texturing
  - Using basic colors
  - Using an image



Exporting the model

- Better Blender x3d exporter:
  - http://www.bitbucket.ca/~acheater/blender/
- Y-up versus Z-up

Convert with J3dFly



- Convert into Wonderland format
  - Loads from x3d
  - Plugin exports to Wonderland local art
  - *Not always WYSWYG*

Place the Model in World

- Wonderland File System (WFS)
  - World layout stored as directories and XML files
  - In development right now

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0" class="java.beans.XMLDecoder">
<!-- the cell type -->
<object class="org.wonderland.ModelViewerCellProperties">

 <!-- cell origin -->
 <void property="cellOrigin">...</void>

 <!-- model file -->
 <void property="fileName">
    <string>models/monkey.j3s.gz</string>
 </void>
```

Update the World

- Reload button in manager UI

Adding Animation...

- Right now done through Maya
- Uses proprietary .rtg format
- Working on open animation pipeline

Agenda

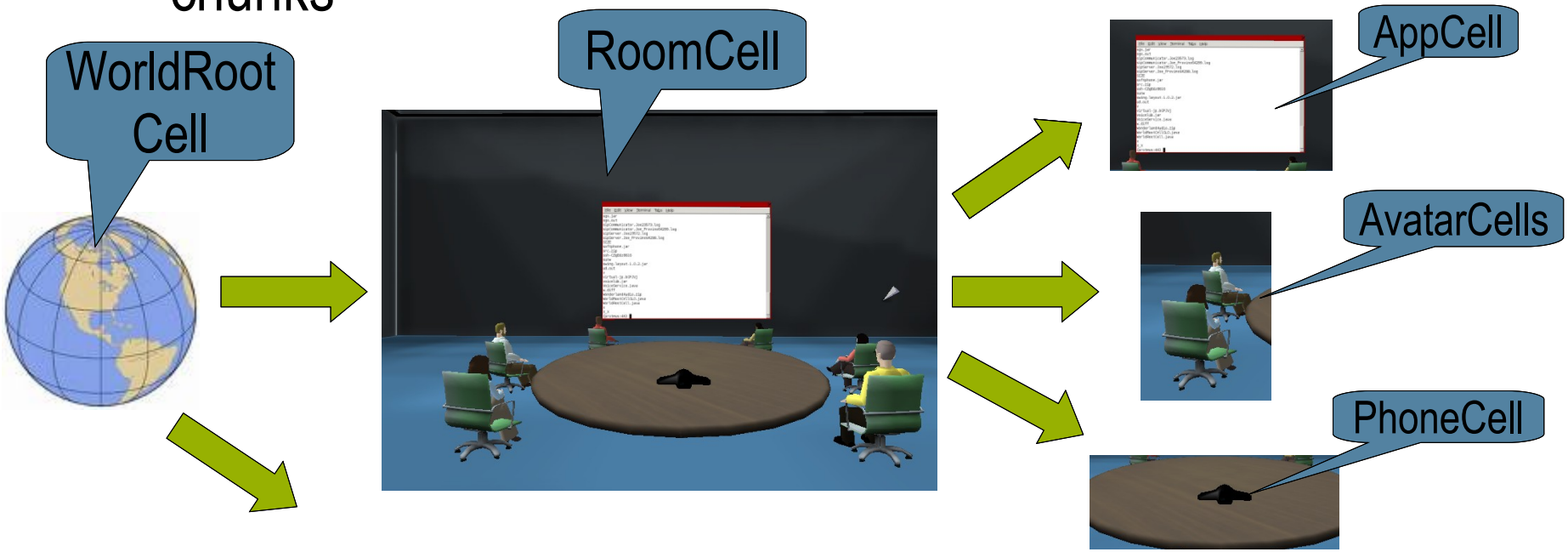
What is Wonderland?
Getting and running Wonderland
Adding artwork
➔ **Programming new features**
Conclusion

Overview of Programming Model

- Wonderland client
  - World layout based on Java3D scene graph
  - Uses Java3D and Project Looking Glass for interactivity
- Wonderland server
  - Darkstar based
  - Manages shared state and channels
  - Controls voice bridge
- Data and communications organized in Cells

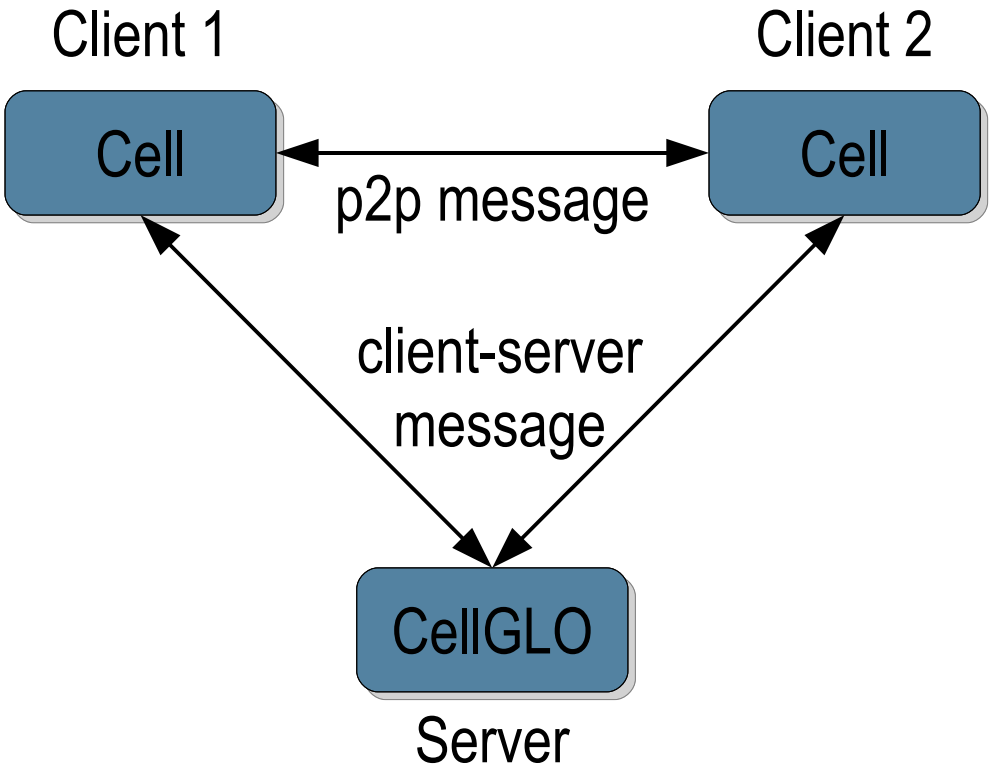
About Cells

- A cell represents a volume of space in the world
  - Cells are nested to build a tree structure
  - Cells divide the Java3D scene graph into network-sized chunks

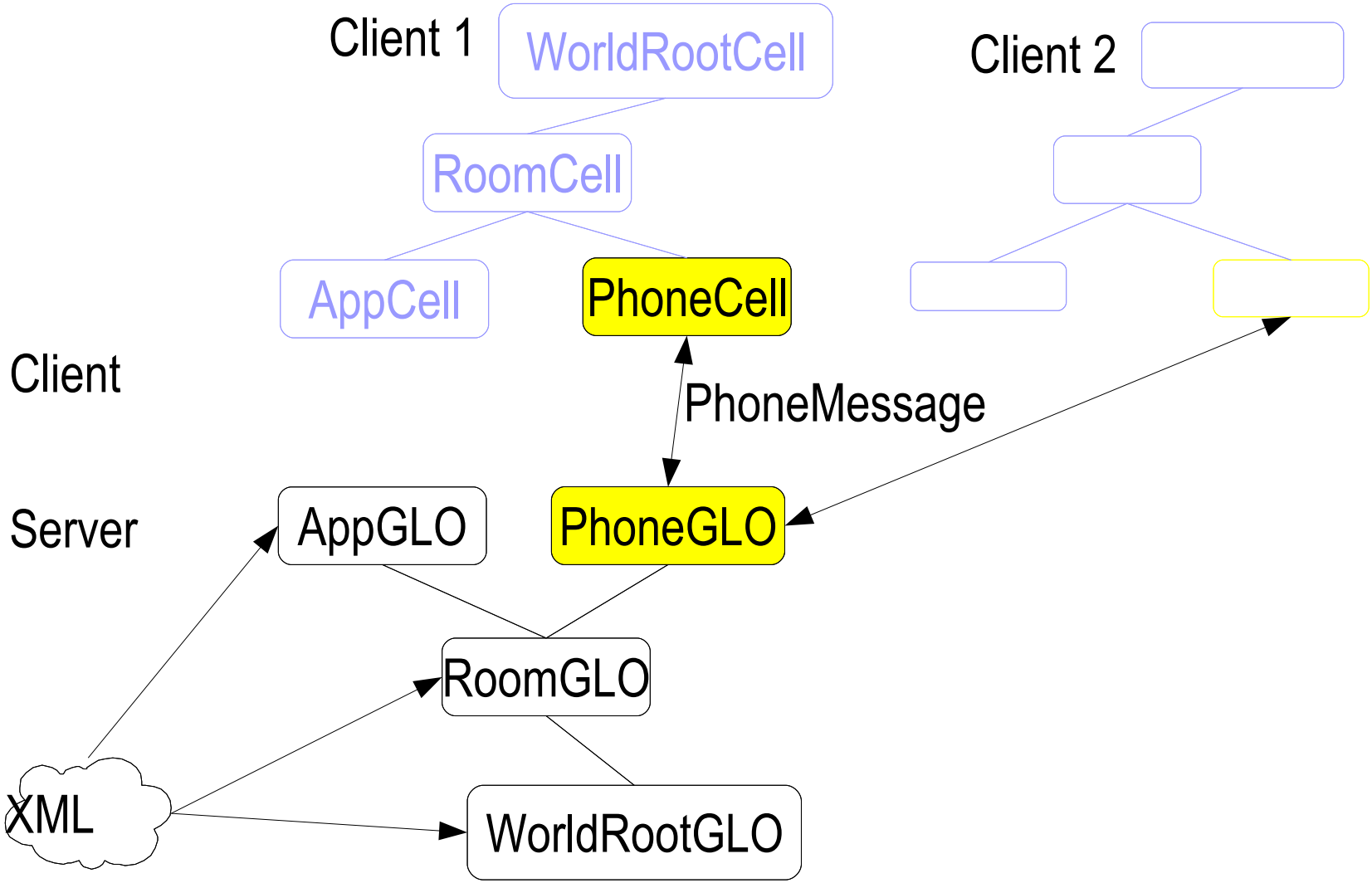


Cells: Client and Server

- Each cell includes a client and server object
- Cells communicate via client-server or p2p messages



Creating a New Cell Type



Client: PhoneCell

- Extends cell
  - Fixed lifecycle
  - Loaded based on server request
- Setup() method
  - Add Java3D geometry
  - Setup mouse listener
- Mouse listener
  - Send a PhoneCellMessage to the server

PhoneCell.java

```
public class PhoneCell extends Cell {
  public void setup(CellSetup setupData) {
    PhoneCellSetup s = (PhoneCellSetup)setupData;

    //Load the model
    BranchGroup phoneBranchGroup =
      AssetManager.getAssetManager().loadGraph(...);

    // Add mouse click handler to the phone branch
    J3dLgBranchGroup bg = new J3dLgBranchGroup();
    bg.addListener(new MouseSelListener());

    //Attach to the Java3D scene graph
    bg.addChild(phoneBranchGroup);
    cellLocal.addChild( bg );
  }
```

PhoneCell.java cont'd

```
// handle mouse clicks
class MouseSelListener implements LgEventListener {
  public void processEvent(LgEvent evt) {
    ChannelController controller =
      ChannelController.getController();

    // send a message to the server to place a call
    PhoneCellMessage msg = new PhoneCellMessage(
          getCellID(), "John Doe", "555-555-5555");
    controller.sendMessage(msg);
  }
}
}
```

Message: PhoneCellMessage

- Extends CellMessage
- Stores state variables
- Manual serialize/deserialize

PhoneCellMessage.java

```
public class PhoneCellMessage extends CellMessage {
  private String name;
  private String phoneNumber;

  // read message from bytes
  protected void extractMessageImpl(ByteBuffer dat) {
    super.extractMessageImpl(dat);
    this.name = DataString.value(dat);
    this.phoneNumber = DataString.value(dat);
  }

  // write data to bytes
  protected void populateDataElements() {
    super.populateDataElements();
    dataElements.add(new DataString(name));
    dataElements.add(new DataString(phoneNumber));
  }
}
```

Server: PhoneCellGLO

- Darkstar managed object
  - Manages the cell channel
  - Persisted in the data store
- Receive message
  - Darkstar task
  - Setup the call
  - Interface with VoiceHandler Darkstar service
  - Send message back to all clients

PhoneCellGLO.java

```
public class PhoneCellGLO extends StationaryCellGLO
    implements CellMessageListener {

  private String modelFile;

  // return the client class for this cell
  public String getClientCellClassName() {
    return "com.sun.labs.phone.client.PhoneCell";
  }

  // get the setup data for the client
  public CellSetup getSetupData() {
    return new PhoneCellSetup(modelFile);
  }
```

PhoneCellGLO.java cont'd

```
  // called when the server receives a message
  public void receivedMessage(ClientSession c,
                              CellMessage m) {

    PhoneCellMessage pcMsg = (PhoneCellMessage) m;

    // create a new call
    VoiceHandler vh = VoiceHandlerImpl.getInstance();
    callID = vh.setupCall(pcMsg.getPhoneNumber(),
                          vh.getVoiceBridge());

    // place the call's audio at the phone's position
    Vector3d translation = new Vector3d();
    this.cellOrigin.get(translation);
    voiceHandler.setPosition(callID, translation.x,
                         translation.y, translation.z);
  }
}
```

Place the cell in world

- Just like adding artwork
- Change the *type* of the cell

```
<java version="1.6.0" class="java.beans.XMLDecoder">
  <!-- the cell type -->
  <object class="com.sun.labs.PhoneCellProperties">

  <!-- cell origin -->
  <void property="cellOrigin">...</void>

  <!-- model file -->
  <void property="fileName">
    <string>models/phone.j3s.gz</string>
  </void>
```

Agenda

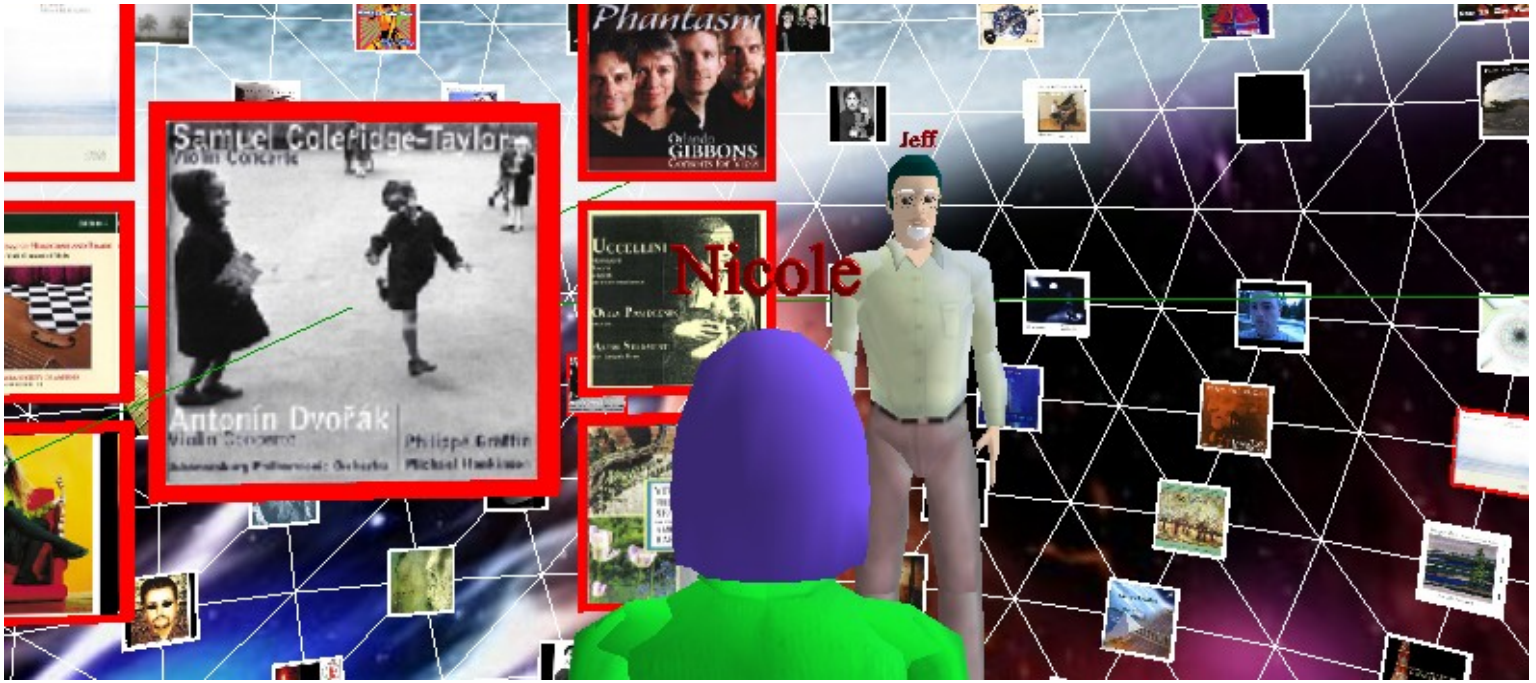
What is Wonderland?
Getting and running Wonderland
Adding artwork
Programming new features
➔ **Conclusion**

Future Work

- New features
  - Heads up display, enhanced telephony, ...
  - More expressive avatars
- New APIs
  - Avatar control, video integration, ...
- Improved art path
  - In-world placement
  - Support new formats: Collada, etc.
  - In-world editing
- Scalability!

Demo: Music in Wonderland

- Collaborative music space
- Built entirely in a single cell
- Integrates music search into the server



Resources

Project Wonderland

http://wonderland.dev.java.net

Project Darkstar

http://www.projectdarkstar.com

MPK20, Sun's Virtual Workplace

http://research.sun.com/projects/mc/mpk20.html