Today’s Class

• Highlights from HW #1
• This Week’s Readings
• Next Week’s Readings

• VTK Graphs
• Intro to Cmake
• Intro to Git

Collision detection: Is it easy to do? Is it necessary?

Do you like VTK’s automatic camera placement? What else does make easy?

Hierarchy of Transformation? Any VTK Tips/Tricks?

Visualization for Debugging: Would this help you study & debug a maze solving algorithm?
How did performance scale with this much geometry?

How do you do textures?

What was most challenging technical detail?

Today’s Class
- Highlights from HW #1
- This Week’s Readings
- Next Week’s Readings
  - VTK Graphs
  - Intro to Cmake
  - Intro to Git

Readings for This Week:
- "Eenie, Meenie, Minie, Moe: Selecting the Right Graph for Your Message" Stephen Few, 2004

Readings for This Week:
- "Designing Effective Step-By-Step Assembly Instructions" Agrawala et al., 2003
Today’s Class
• Highlights from HW #1
• This Week’s Readings
• Next Week’s Readings

• VTK Graphs
• Intro to Cmake
• Intro to Git

Readings for Next Week:
• “Graph drawing by force-directed placement”, Fruchterman & Reingold, 1991

Readings for Next Week:
• “Heapviz: Interactive Heap Visualization for Program Understanding and Debugging” Aftandilian, Kelley, Gramazio, Ricci, Su, & Guyer, 2010
Types of Graphs

- Directed
  - One vertex "points" to another
- Undirected
  - Two vertices are connected (they both point to each other)

Graphs in VTK

http://www.vtk.org/Wiki/VTK/Examples/Cxx/Graphs

Directed
- One vertex "points" to another

Undirected
- Two vertices are connected (they both point to each other)

Constructing Graphs

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/Graphs/VisualizeGraph
- Use these to construct graphs
  - vtkMutableUndirectedGraph
  - vtkMutableDirectedGraph

```
vtkSmartPointer<vtkMutableUndirectedGraph> g =
vtkSmartPointer<vtkMutableUndirectedGraph>::New();
vtkIdType v1 = g->AddVertex();
vtkIdType v2 = g->AddVertex();
g->AddEdge(v1, v2);
```

Graph Data

- VertexData
  - You can store data for every vertex
  - Colors, id, names, etc
  - graph->GetVertexData()->AddArray(yourFavoriteArray);
- EdgeData
  - You can store data for every edge
  - Edge weights, colors, etc
  - graph->GetEdgeData()->AddArray(yourFavoriteArray);
  - http://www.vtk.org/Wiki/VTK/Examples/Cxx/Graphs/EdgeWeights

```
... Create a graph with 3 vertices and 3 edges ...
```

```
// Create the edge weight array
vtkSmartPointer<vtkDoubleArray> weights =
vtkSmartPointer<vtkDoubleArray>::New();
weights->SetNumberOfComponents(1);
weights->SetName("Weights");
// Set the edge weights
weights->InsertNextValue(1.0);
weights->InsertNextValue(1.0);
weights->InsertNextValue(2.0);

// Create the vertex id array
vtkSmartPointer<vtkIntArray> vertexIDs =
vtkSmartPointer<vtkIntArray>::New();
vertexIDs->SetNumberOfComponents(1);
vertexIDs->SetName("VertexIDs");
// Set the vertex ids
vertexIDs->InsertNextValue(0);
vertexIDs->InsertNextValue(1);
vertexIDs->InsertNextValue(2);

// Add the edge weight array to the graph
graph->GetEdgeData()->AddArray(weights);
graph->GetVertexData()->AddArray(vertexIDs);
```

Graph Data: Example

```
... Create a graph with 3 vertices and 3 edges ...
```

```
// Create the edge weight array
vtkSmartPointer<vtkDoubleArray> weights =
vtkSmartPointer<vtkDoubleArray>::New();
weights->SetNumberOfComponents(1);
weights->SetName("Weights");
// Set the edge weights
weights->InsertNextValue(1.0);
weights->InsertNextValue(1.0);
weights->InsertNextValue(2.0);

// Create the vertex id array
vtkSmartPointer<vtkIntArray> vertexIDs =
vtkSmartPointer<vtkIntArray>::New();
vertexIDs->SetNumberOfComponents(1);
vertexIDs->SetName("VertexIDs");
// Set the vertex ids
vertexIDs->InsertNextValue(0);
vertexIDs->InsertNextValue(1);
vertexIDs->InsertNextValue(2);

// Add the edge weight array to the graph
graph->GetEdgeData()->AddArray(weights);
graph->GetVertexData()->AddArray(vertexIDs);
```

Displaying Graphs

- http://www.vtk.org/Wiki/VTK/Examples/Cxx/Graphs/VisualizeGraph

```
vtkSmartPointer<vtkGraphLayoutView> graphLayoutView =
vtkSmartPointer<vtkGraphLayoutView>::New();
graphLayoutView->AddRepresentationFromInput(g);
graphLayoutView->ResetCamera();
graphLayoutView->Render();
graphLayoutView->GetInteractor()->Start();
```
Displaying Vertex and Edge Labels

```
graphLayoutView->SetVertexLabelVisibility(true);
graphLayoutView->SetEdgeLabelVisibility(true);
graphLayoutView->SetEdgeLabelArrayName("Weights");
```

- \( \text{graphLayoutView} \rightarrow \text{SetVertexLabelVisibility} \) (true);
- \( \text{graphLayoutView} \rightarrow \text{SetEdgeLabelVisibility} \) (true);
- \( \text{graphLayoutView} \rightarrow \text{SetEdgeLabelArrayName} \) ("Weights");

Layout Strategies

- \( \text{graphLayoutView} \rightarrow \text{SetLayoutStrategy} \) ("Strategy Name");
  - "Random": Randomly places vertices in a box
  - "Force Directed": Simulating forces on edges
  - "Simple 2D": A simple 2D force directed layout
  - "Clustering 2D": Just like simple 2D but uses some techniques to cluster better
  - "Fast 2D": A linear-time 2D layout.
  - "Circular"

You can also specify the "Pass Through" layout strategy

This allows you to set the coordinates of the vertices manually:

```vpython
vtkSmartPointer<vtkPoints> points = vtkSmartPointer<vtkPoints>::New();
points->InsertNextPoint(0.0, 0.0, 0.0);
points->InsertNextPoint(1.0, 0.0, 0.0);
points->InsertNextPoint(0.0, 1.0, 0.0);
g->SetPoints(points);
```

Neighboring Vertices

```
vtkAdjacentVertexIterator iterator = vtkSmartPointer<vtkAdjacentVertexIterator>::New();
g->GetAdjacentVertices(0, iterator);
while(iterator->HasNext())
{
  vtkIdType nextVertex = iterator->Next();
  std::cout << "Next vertex: " << nextVertex << std::endl;
}
```

Vertex Coordinates

- You can also specify the "Pass Through" layout strategy

- This allows you to set the coordinates of the vertices manually:

```
vtkSmartPointer<vtkPoints> points = vtkSmartPointer<vtkPoints>::New();
points->InsertNextPoint(0.0, 0.0, 0.0);
points->InsertNextPoint(1.0, 0.0, 0.0);
points->InsertNextPoint(0.0, 1.0, 0.0);
g->SetPoints(points);
```

Lookup Tables

- Automatic map from a range to all colors:

```
  vtkSmartPointer<vtkLookupTable> lookupTable = vtkSmartPointer<vtkLookupTable>::New();
  lookupTable->SetTableRange(0.0, 10.0);
  lookupTable->Build();
```

- Manually specify colors in the table:

```
  vtkSmartPointer<vtkLookupTable> lookupTable = vtkSmartPointer<vtkLookupTable>::New();
  lookupTable->SetNumberOfTableValues(2);
  lookupTable->SetTableValue(0, 1.0, 0.0, 0.0); // red
  lookupTable->SetTableValue(1, 0.0, 1.0, 0.0); // green
  lookupTable->Build();
```

Color Vertices

```
  // Create the color array
  vtkIntArray* vertexColors = vtkIntArray::New();
  vertexColors->SetName("VertexColors");
  vertexColors->SetNumberOfComponents(1);
  vertexColors->InsertNextValue(1);
  vertexColors->InsertNextValue(2);
  ...
  // Add the color array to the graph
  graph->GetVertexData()->AddArray(vertexColors);
```

```
  // Create a lookup table ...
```

- Add the color array to the graph:

```
  // Create the color array
  vtkIntArray* vertexColors = vtkIntArray::New();
  vertexColors->SetName("VertexColors");
  vertexColors->SetNumberOfComponents(1);
  vertexColors->InsertNextValue(1);
  vertexColors->InsertNextValue(2);
  ...
  // Add the color array to the graph
  graph->GetVertexData()->AddArray(vertexColors);
```

```
  // Create a lookup table ...
```

```
  // Create the color array
  vtkIntArray* vertexColors = vtkIntArray::New();
  vertexColors->SetName("VertexColors");
  vertexColors->SetNumberOfComponents(1);
  vertexColors->InsertNextValue(1);
  vertexColors->InsertNextValue(2);
  ...
  // Add the color array to the graph
  graph->GetVertexData()->AddArray(vertexColors);
```

```
  // Create a lookup table ...
```

- Add the color array to the graph:

```
  // Create the color array
  vtkIntArray* vertexColors = vtkIntArray::New();
  vertexColors->SetName("VertexColors");
  vertexColors->SetNumberOfComponents(1);
  vertexColors->InsertNextValue(1);
  vertexColors->InsertNextValue(2);
  ...
  // Add the color array to the graph
  graph->GetVertexData()->AddArray(vertexColors);
```

```
  // Create a lookup table ...
```
Today’s Class

• Highlights from HW #1
• This Week’s Readings
• Next Week’s Readings

• VTK Graphs
  • Intro to CMake
  • Intro to Git

What is CMake?

• Cross platform Make

• Allows the same source code to be compiled on many different operating systems and IDEs (Integrated Development Environment)
  o Visual Studio
  o Code Blocks
  o Eclipse
  o Traditional Unix Makefile
  o etc.

Single Person Project

The Code

The Compiler/IDE

The Idea

• Software used to ship with a Makefile

• The settings for library paths, which options you wanted to use for the compilation and installation were hard coded into the Makefile

• Autoconf and similar packages attempted to make this a little easier

• CMake makes it VERY easy

Introduction to VTK: CMake

ColorEdges

// Create the color array
vtkIntArray *edgeColors = vtkIntArray::New();
edgeColors->SetName("EdgeColors");
edgeColors->SetNumberOfComponents(1);
edgeColors->InsertNextValue(1);
edgeColors->InsertNextValue(2);

... Create a lookup table...

// Add the color array to the graph
graph->GetEdgeData()->AddArray(edgeColors);

vtkViewTheme *theme = vtkViewTheme::New();
theme->SetCellLookupTable(lookupTable);
graphLayoutView->ApplyViewTheme(theme);
graphLayoutView->SetVertexColorArrayName("EdgeColors");
graphLayoutView->ColorEdgesOn();

Today’s Class

• Highlights from HW #1
• This Week’s Readings
• Next Week’s Readings

• VTK Graphs
  • Intro to CMake
  • Intro to Git

What is CMake?

• Cross platform Make

• Allows the same source code to be compiled on many different operating systems and IDEs (Integrated Development Environment)
  o Visual Studio
  o Code Blocks
  o Eclipse
  o Traditional Unix Makefile
  o etc.

Single Person Project

The Code

The Compiler/IDE

The Idea

• Software used to ship with a Makefile

• The settings for library paths, which options you wanted to use for the compilation and installation were hard coded into the Makefile

• Autoconf and similar packages attempted to make this a little easier

• CMake makes it VERY easy

Introduction to VTK: CMake

ColorEdges

// Create the color array
vtkIntArray *edgeColors = vtkIntArray::New();
edgeColors->SetName("EdgeColors");
edgeColors->SetNumberOfComponents(1);
edgeColors->InsertNextValue(1);
edgeColors->InsertNextValue(2);

... Create a lookup table...

// Add the color array to the graph
graph->GetEdgeData()->AddArray(edgeColors);

vtkViewTheme *theme = vtkViewTheme::New();
theme->SetCellLookupTable(lookupTable);
graphLayoutView->ApplyViewTheme(theme);
graphLayoutView->SetVertexColorArrayName("EdgeColors");
graphLayoutView->ColorEdgesOn();
The Magic File — CMakeLists.txt

```
PROJECT(YourExampleProject)
FIND_PACKAGE(VTK REQUIRED)
INCLUDE(VTK_USE_FILE)
ADD_EXECUTABLE(ExampleExecutable ExampleCode.cxx)
TARGET_LINK_LIBRARIES(ExampleExecutable vtkHybrid)
```

Line-by-line

1. PROJECT(YourExampleProject)
2. Tell CMake you are starting a new project
3. FIND_PACKAGE(VTK REQUIRED)
4. Tell CMake you want to use VTK in your new project
5. INCLUDE(VTK_USE_FILE)
6. If you have the environment variable VTK_BIN set, CMake will find VTK automatically
7. If not, when you configure your project using CMake it will ask you where you have VTK installed
8. ADD_EXECUTABLE(ExampleExecutable ExampleCode.cxx)
9. Tell CMake you want to create an executable called ExampleExecutable from ExampleCode.cxx
10. TARGET_LINK_LIBRARIES(ExampleExecutable vtkHybrid)
11. Tell CMake that ExampleCode.cxx uses functions defined in the library vtkHybrid. You can list as many libraries as you need.
Windows Interface

- Download from here: http://www.cmake.org/cmake/resources/software.html
- You will use a GUI interface

Windows Process

- Run CMake
- Point it to your source directory
- Point it the build directory of your choice
- Set an options that you would like
- Click “configure” (you may have to do this twice (until the “generate” button is not greyed out))
- Choose your generator (which IDE you are going to use to build the project)
- Click “Generate”

Linux

- Should already have CMake installed
- Test by typing ‘cmake’ in a terminal
- If you don’t get “command not found”, you’re in good shape
- If you do, most distributions have a CMake package
  - `sudo yum install cmake`
  - `sudo apt-get install cmake`
  - Or equivalent

Linux Interface

- `ccmake` – Curses Cmake (http://en.wikipedia.org/wiki/Curses_(programming_library))
- Allows you to set many options before generating a project

Linux Process

- Linux - Makefile
  - Create a build directory (wherever you want)
  - From the build directory, run ‘ccmake’ on the source directory
  - Example:
    - Your source code is in ~/src/VTK
    - Create a build directory: mkdir ~/bin/VTK
    - cd ~/bin/VTK
    - cmake ~/src/VTK
  - Set any options you would like
  - Hit ‘c’ for ‘configure’. You may have to do this twice
  - Hit ‘g’ for ‘generate’

Linux Process

- Linux – Other IDE
  - `ccmake ~/src/VTK` –G YourGenerator
  - More info at http://www.vtk.org/Wiki/CMake_Generator_Specific_Information
  - Generators include
    - Eclipse
    - KDevelop3
    - CodeBlocks
    - NMake
First Time VTK Programming

- Step 1 – Download the VTK source code
- Step 2 – Use CMake to create a project for the IDE you want to use to compile VTK
- Step 3 – Use the IDE you selected in Step 2 to build VTK
- Step 4 – Obtain an example program and associated CMakeLists.txt file from the Examples Wiki
- Step 5 – Use CMake to create a project for the IDE you want to use to compile the example program
- Step 6 – Use the IDE you selected in Step 5 to build the example program

Today’s Class

- Highlights from HW #1
- This Week’s Readings
- Next Week’s Readings
  - VTK Graphs
  - Intro to Cmake
  - Intro to Git

Introduction to VTK: Git

What is Git?

- The latest and greatest version control system
- Makes large projects easier to manage
- Written by Linus Torvalds (the Linux guy) to help with the massive Linux project

Version Control Systems – A Brief History

- Patches
- CVS/SVN
- Git

The Olden Days

- In the “olden days”, people would send patches back and forth to each other via email
- This was terribly inefficient
- The patch would only work with the exact file that it was created for
- No version control
**CVS/SVN**

- CVS: Concurrent Versions System
- SVN: Subversion – a re-write of CVS
- Users can “publish” code to a central repository
- Users can get the code from the central repository

**Git**

- There is no “central repository” to speak of
- Typically one computer is designated as the “official” computer, but it is no different than any other user

**Branches**

- Work on different “projects” without affecting other “projects” in the works
- Submit a branch for easy code review
- Github.com gitorious.com, etc.