

Files, Widgets & Coding Rules

COMP 4004
Dr. Hamish Carr
A1.01
01 716 2475
hamish.carr@ucd.ie



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Coding Rules

- *Code* must be:
 - your *own* work, except *support* code:
 - a) code from 4004 *only* (not 3003)
 - b) free cross platform libraries



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Data Rules

- *Data* must be:
 - a) your own work, or
 - b) freely available
 - i.e. *with* copyright permission



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Evaluation Rules

- Assignments will have:
 - *required* part (70%)
 - *optional* part (30%)
- Optional *may* include your own ideas
 - at *lecturer's* discretion



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Software Bounty

- There's lots of useful code out there
- I don't know all of it
- *Small* (1%) bounty for *useful* software, data or libraries for class
- Again, at *lecturer's* discretion



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

C++ Objects

- C++ is natural for mathematics
 - classes = new data types
 - operator overloading
 - but there's a problem
 - *anonymous* variables



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Anonymous Variables

- $z = x^2 + y;$
- Where is (x^2) stored?
- Compiler allocates storage (or register)
 - in an *anonymous* variable
- And cleans up afterwards



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Operator Overloading

- What is $(x*z)$? (cross-product)
- Where is $(x*z)$ stored? $\text{Vector } w, x, y, z;$
 $y = x * z + w;$
- operator returns a *reference* to an object
 - but *not* an anonymous object
- *Only primitive types can be anonymous*



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Solution

```
// Copy another Point & return self
Point &Point::Assign(Point &other)
{ // Assign()
  for (int i = 0; i < 4; i++)
    coords[i] = other.coords[i];
  return (*this);
} // Assign()
```

- Function *behaves* like an operator
- Applies operation to variable
- And passes variable on



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Result:

```
Matrix A;
A.Assign(B).RightMultiply(C);
```

- Applies operation *to* parameter A
- Returns reference
- Applies next operation
- Performs this:

```
Matrix A;
A = B * C;
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Point class

- Stores homogeneous coordinates
 - less efficient on bandwidth
 - but convenient
 - can be used in `glVertex4fv()`;



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Point operations

```
// Copy another Point & return self
Point &Assign(Point &other);

// Multiply by a constant & return self
Point &Scale(GLfloat scaleFactor);

// Divide through by homogeneous coordinate and return self
Point &DeHomogenize();

// Adds a vector in Cartesian space
Point &AddVector(Vector &vector);

// returns 0 on success, non-0 on failure
long ReadCartesian(FILE *inFile);

// routines to print it out
void Print(FILE *outFile);
void PrintCartesian(FILE *outFile);
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Vector class

- Stores Cartesian coordinates
- w coordinate not *needed*

```
// Copy another Vector & return self
Vector &Assign(Vector &other);

// Multiply by a constant & return self
Vector &Scale(GLfloat scaleFactor);

// Computes the length (sqrt of sum of squares)
GLfloat Length();

// Normalize and return self
Vector &Normalize();
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Vector operations

```
// Add another Vector & return self
Vector &Add(Vector &other);

// Subtract another Vector & return self
Vector &Subtract(Vector &other);

// Subtract two points to get a vector: return self
Vector &Between(Point &from, Point &to);

// Compute the cross-product (this x other) & return self
Vector &Cross(Vector &other);

// Dot product computes (this . other)
GLfloat Dot(Vector &other);

// returns 0 on success, non-0 on failure
long ReadCartesian(FILE *inFile);

// routine to print it out
void Print(FILE *outFile);
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Matrix class

- Stored as 16 floats OpenGL style
- single flat array
- in *column-major* order
- Coded for *clarity*, not *efficiency*



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Matrix operations

```
// set the matrix to the identity matrix
Matrix &Identity();

// set the matrix to the zero matrix
Matrix &Zero();

// returns a specified element in the matrix
GLfloat &Element(int row, int col);

// Copy another Matrix & return self
Matrix &Assign(Matrix &otherMatrix);

// Right-add another Matrix & return self
Matrix &Add(Matrix &otherMatrix);

// Scale the matrix by a constant & return self
Matrix &Scale(GLfloat scaleFactor);
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Matrix operations

```
// Left-multiply a Point, return the Point
Point &LeftMultiply(Point &point);

// Left-multiply a Vector, return the Vector
Vector &LeftMultiply(Vector &vector);

// Right-multiply by another matrix & return self
Matrix &RightMultiply(Matrix &otherMatrix);

// routine to read it in
// returns 0 on success, non-0 on failure
long Read(FILE *inFile);

// routine to print it out
void Print(FILE *outFile);
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Matrix operations

```
// Scales a row by a given factor
void ScaleRow(int row, GLfloat scaleFactor);

// Swaps two rows
void SwapRows(int row1, int row2);

// Adds multiple of one row to another row
void AddRows(int row1, GLfloat multiple, int row2);

// Reduces non-diagonal elements in a column
// Returns 0 on success, 1 if the column cannot be reduced
int ReduceColumn(int column, Matrix &inverse);

// Computes inverse with brute force Gauss-Jordan method
// Returns 0 on success, non-0 on failure
int Invert();
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

File Issues

- ASCII vs. binary
- Big-endian vs. little-endian
- Mesh & Animation
- Hierarchical vs. Flat
- Lex / Yacc



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

ASCII vs. binary

- ASCII text is:
 - easy to read (humans)
 - slow to read (computers)
- Binary is:
 - fast to read (computers)
 - hard to read (humans)



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Endianness

- Little-endian: 4321 (array-style)
 - Intel chips
- Big-endian: 1234 (human-style)
 - Almost everyone else
- *And* this applies to data files
- So stick to text files if you can



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Image File Formats

- PBM / PGM / PPM / RAW
 - Simple (dumps the pixel array), large
 - I'll provide code for this
- JPG/TIFF/GIF/&c.
 - Complicated
- Some libraries exist (e.g. libjpeg)



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Mesh File Formats

- Common modelling formats:
 - Alias/Wavefront Maya (.obj)
 - 3D Studio Max (.3ds)
 - VRML (.vrm)l
 - Blender (multiple formats)
- Free models at www.3dcafe.com



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Animation File Formats

- Modelling formats support animation
 - But we need something *simple*
- BVH files (.bvh)
 - Simple text format
- Free models at www.animazoo.com



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Scene Graphs

- Modelling files store *scene graphs*
 - animation hierarchy for entire scene
 - stored in a tree in memory
 - stored to disk by tree traversal
 - tricky to read, but versatile



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Hierarchical vs. Flat Files

- *Hierarchical* files are complicated
 - *hard to load* (.3ds, &c)
- *Flat* files (i.e. arrays) are easier
 - *simple* arrays in memory & on disk
- *For teaching purposes, we'll use flat files*



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Lex / Yacc

- Lex & yacc are compiler tools
 - used to parse files
 - often used for hierarchical files
- But we won't worry about this



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

lib3ds

- Open source code for .3ds files
- Reads in a file in hierarchical form
- But it's messy and complex
- So we will use it to *convert* files



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Mesh Conversion Path

- Convert to .3ds first
- Convert to .m with *3ds2m* (lib3ds)
- I've added some extras to get .m2
- I'll distribute modified 3ds2m code



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Coding Platforms

- Linux (lab machines)
- OS X (anyone with a Mac)
- Windows (anyone?)



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Linux

- OpenGL or Mesa should be installed
- GLUT may need installation
- Toolkits / libraries will need installation
- Compilation / makefiles are straightforward



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

OS X

- OS X has non-standard locations
 - and also strange compile flags
 - ask me for a makefile
- But OpenGL / GLUT are installed
 - with the developer tools



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Windows

- First problem: compiler
- Second problem: library location
- Third problem: library linking
- Fourth problem: GLUT installation
- Linux is a better choice on x86



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

GUI Widgets

- Last year, we used GLUT
 - easy to install
 - handles *simple* tasks easily
- This year, you will want *widgets*
 - a proper GUI
 - miscellaneous support code



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

GUI Review

- Event Loops
- Event Types
- Event Processing
- Widgets
- The Visual Hierarchy
- Toolkits



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Events

- GUIs are based on *events*
 - actions by the *user*
- The program *waits* for the user
 - then responds to the user's action



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Event Loop

- The *event loop* is a simple loop:

```
while (not done)           while (not done)
{
  CheckForMouse();         { event = system.GetEvent();
  CheckForKeyboard();      ProcessEvent(event);
  CheckSystemEvents();     UpdateDisplay();
  ProcessEvents();         }
  UpdateDisplay();
}
```

- Just keeps checking user input
- gets very complicated



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

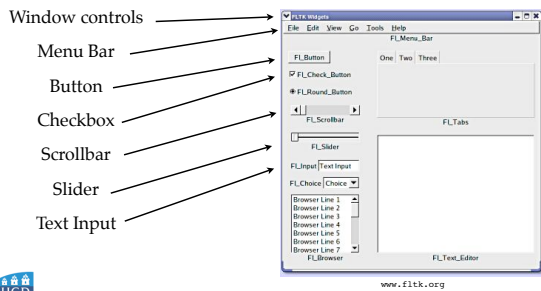
Windows & Widgets

- GUIs interact through *windows*
 - *visual* representations of data
 - with visual *controls* for manipulation
- Windows are divided into *widgets*
 - visual entities in the window



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Sample Widgets



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Event Types

- Three fundamental types:
 - *mouse* events
 - *keyboard* events
 - *system* events
- And various *derived* types



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Mouse Events

- Mouse buttons can be:
 - *pressed* (initially)
 - *dragged* (at multiple positions)
 - *released* (finally)
- Display should *reflect* events *visually*



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Drawing a Line

- On mouse down, create temporary line
 - both ends at location of mouse
- On mouse drag, update one end
- On mouse up, *commit* the line to memory
 - i.e. add it *permanently*
- This is also called *rubber-banding*



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Comments

- Rubberbanding:
 - gives visual *feedback*
 - allows the user *control*
 - and is *reversible*
- All of which are good habits



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Keyboard Events

- Much easier than mouse events
 - User presses one or more keys
 - But the processing can be *complex*



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

System Events

- Idle processing (periodic)
 - when no other events to process
- Incoming network events
 - &c., &c., &c.



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Derived Events

- Widgets have standard semantics
- Many widgets are *controls*
 - actions influence a single variable
 - indirectly influence application
- Derived events handle this for you



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Button Presses

- Pressing a button can:
 - *be processed manually by the coder*
 - *update a variable (live variables)*
 - *invoke a function*
 - *send a message*
- Still basically a mouse event



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Toolkit Paradigms

- Direct Event Processing (Mac OS 1)
- Callback Functions (X / GLUT / GLUI)
- Live Variables (GLUI)
- Hierarchical Event Processing (Fltk)
- Virtual Function Event Processing (Qt)



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Callback Processing

- Based on *C function pointers*
 - a pointer to a function in memory
 - called by dereference: `(*myFunc)(x)`
 - can be dangerous / hackable
- Pass a *callback* function for each event



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Callback Processing

```
while (not done)
{ // event loop
  Event e = system.getEvent();
  switch(e.type)
  { // event type
    NULL:      (*idle)(e);      break;
    MOUSE:     (*mouse)(e);     break;
    KEYBOARD:  (*keyboard)(e);  break;
    SYSTEM:    (*system)(e);    break;
  } // event type
  (*display)();
} // event loop
```



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Widget Focus

- Each widget processes events differently
 - we need to decide *which widget*
 - based on *focus* - i.e. user's intention
 - *focus follows mouse* - X standard
 - *sticky focus* - Mac / Windows



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Focus Follows Mouse

- All events happen under cursor
 - *including* keyboard events
 - disconcerting if mouse jiggles
 - arguably more powerful



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Sticky Focus

- Mouse events occur at mouse location
- Keyboard events occur in *active widget*
 - i.e. last widget *chosen* by user
 - arguably easier for user



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Event Location

- In either case, events have a *location*
 - a cursor location, or
 - a specific window or widget
- How do we *find* the right widget?



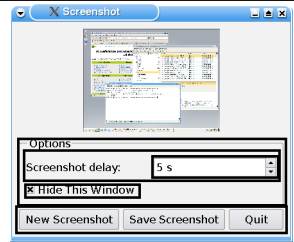
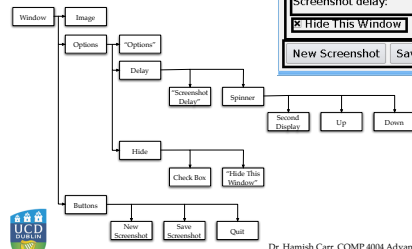
Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

The Visual Hierarchy

- Widgets are *inside* each other
- They form a natural hierarchy
 - *visual hierarchy*
 - tree structure
- Often used for *event dispatch*



Visual Hierarchy



Visual Layout

- Each group is *horizontal* or *vertical*
 - how children are laid out visually
- What is the layout of the window?
- Positions can be given directly
 - or left up to a *layout manager*



Hierarchical Processing

```
while (not done)
{ // event loop
  Event e = system.getEvent();
  rootWindow->ProcessEvent(e);
  rootWindow->Redraw();
} // event loop
```

The Main Event Loop

```
Window::ProcessEvent(Event e)
{ // event loop
  for (int child = 0; child < nChildren; child++)
    if (children[child].contains(e.location))
      children[child]->ProcessEvent(e);
} // event loop
```

Hierarchical Processing (Recursive)

Fltk Event Processing

```
int MyClass::handle(int event) {
  switch(event) {
    case FL_PUSH:
      highlight = 1;
      redraw();
      return 1;
    case FL_DRAG: {
      int t = Fl::event_inside(this);
      if (t != highlight) {
        highlight = t;
        redraw();
      }
    }
    default:
      return Fl_Widget::handle(event);
  }
}
```

Hierarchical, but all in one function



Qt Event Processing

```
void GLWidget::mousePressEvent(QMouseEvent *event)
{
  lastPos = event->pos();
}

void GLWidget::mouseMoveEvent(QMouseEvent *event)
{
  int dx = event->x() - lastPos.x();
  int dy = event->y() - lastPos.y();

  if (event->buttons() & Qt::LeftButton) {
    setXRotation(xRot + 8 * dy);
    setZRotation(yRot + 8 * dx);
  } else if (event->buttons() & Qt::RightButton) {
    setXRotation(xRot + 8 * dy);
    setZRotation(zRot + 8 * dx);
  }

  lastPos = event->pos();
}
```

Hierarchical, in separate functions



GLUI

- 3D controls (e.g. arcball)
- Live variables
- Simple to add to GLUT
- Limited set of widgets
 - e.g. no file dialogs
- Single theme (old X-Windows)



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Fltk

- Wide range of widgets
- Single theme (old X-Windows)
- No 3D-specific controls
- Event processing all in one function



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006

Qt

- Wider range of widgets
- Commercial, so well-supported
- Uses OS-native widgets
- Simpler event processing
- Somewhat trickier install
- Limited 3D controls



Dr. Hamish Carr, COMP 4004 Advanced Computer Graphics, Spring 2006