

Introduction to OpenGL Programming – 2D Graphics

Motivation

- We won't touch the low levels of rasterization
 - rely on the GPU to perform scan conversion, etc
- there are a lot of different GPUs out there
 - different brands: ATI, NVIDIA, etc
 - different capabilities
- need standard way of interfacing with GPU
 - send vertices, normals, lights, cameras to GPU
 - wait for hardware to do its magic
 - get the rendered image back
- this is where OpenGL fits in



Middle Tennessee State University

What is OpenGL?

- The Open Graphics Library
 - 3-D graphics API specification
 - raster graphics library
 - pass in vertices, normals, and other scene data
 - get pixels out
 - industry standard
 - specification publicly available
 - supported across many platforms
 - Mac OS, Windows, Linux, iPhone, PSP...

Middle Tennessee State University

What is OpenGL?

- OpenGL is a software API to graphics hardware.
 - designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms
 - Intuitive, procedural interface with C, C++, Java, Perl, Python, ... bindings
 - No windowing commands!
 - No high-level commands for describing models of three-dimensional objects

Middle Tennessee State University

What Is OpenGL?

- A software interface to graphics hardware.
- The interface consists of about 250 commands (functions) to specify the objects and operations needed to produce 2D and 3D graphics
 - OpenGL geometric primitives include points, lines, polylines, and polygons. There is specific support for triangle and quadrilateral
 - Has texture mapping support.

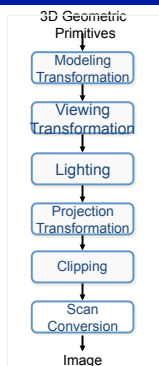
Middle Tennessee State University

OpenGL Libraries

- OpenGL core library
 - OpenGL32 on Windows
 - GL on most unix/linux systems
- OpenGL Utility Library (GLU)
 - Provides functionality in OpenGL core but avoids having to rewrite code
- OpenGL Utility Toolkit (GLUT)
 - Provides functionality common to all window systems
 - Open a window
 - Get input from mouse and keyboard
 - Menus
 - Event-driven
 - Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform
 - No slide bars

Middle Tennessee State University

Classic Rendering Pipeline



Middle Tennessee State University

OpenGL Architecture

- OpenGL uses a client-server model
 - client sends commands to the server
 - server interprets, processes commands
 - note: client and server usually on the same computer, but need not be
 - your program = client
 - OpenGL/GPU = server
- example interaction:

program	OpenGL/GPU
begin triangle normal (0, 0, -1) vertex (-1, 1, -1, 1) vertex (1, -1, -1, 1) vertex (-1, -1, -1, 1) end triangle	<scan converts the given triangle with normal (0,0,-1) on all vertices>

Middle Tennessee State University

OpenGL as a state machine

- **Put OpenGL into states (modes)**
 - Projection and viewing matrix
 - Color and material properties
 - Lights and shading
 - Line and polygon drawing modes
 - ...
- **GL State Variables- can be set and queried by OpenGL. Remains unchanged until the next change.**
- **OpenGL functions are of two types**
 - Primitive generating
 - Can cause output if primitive is visible
 - How vertices are processed and appearance of primitive are controlled by the state
 - State changing
 - Transformation functions
 - Attribute functions

Middle Tennessee State University

OpenGL Syntax

- functions have prefix **gl** and initial capital letters for each word
 - `glClearColor()`, `glEnable()`, `glPushMatrix()` ...
- **glu** for **GLU** functions
 - `gluLookAt()`, `gluPerspective()` ...
- constants begin with **GL_**, use all capital letters
 - `GL_COLOR_BUFFER_BIT`, `GL_PROJECTION`, `GL_MODELVIEW` ...
- Extra letters in some commands indicate the number and type of variables
 - `glColor3f()`, `glVertex3f()` ...
- OpenGL data types
 - `GLfloat`, `GLdouble`, `GLint`, `GLenum`, ...

Middle Tennessee State University

OpenGL function format

function name dimensions

`glVertex3f(x, y, z)`

belongs to GL library `x, y, z` are floats

`glVertex3fv(p)`

`p` is a pointer to an array

Middle Tennessee State University

Open-GL Data Types

suffix	data type	C/C++ type	OpenGL type name
b	8-bit integer	signed char	GLbyte
s	16-bit integer	Short	GLshort
i	32-bit integer	int or long	GLint, GLsizei
f	32-bit float	Float	GLfloat, GLclampf
d	64-bit float	Double	GLdouble, GLclampd
ub	8-bit unsigned number	unsigned char	GLubyte, GLboolean
us	16-bit unsigned number	unsigned short	GLushort
ui	32-bit unsigned number	unsigned int or unsigned long	GLuint, GLenum, GLbitfield

Middle Tennessee State University

OpenGL Syntax Examples

Example: Setting the current color using `glColor`.

- Colors may have 3 components (RGB) or 4 components (RGBA). Think of A (or alpha) as opacity.
- Floating point - color component values range from 0 to 1

```
glColor3f(0.0, 0.5, 1.0);  
This is 0% Red, 50% Green, 100% Blue;  
glColor4f(0.0, 0.5, 1.0, 0.3);  
This is 0% Red, 50% Green, 100% Blue, 30% Opacity  
GLfloat color[4] = { 0.0, 0.5, 1.0, 0.3 };  
glColor4fv(color);  
Again, 0% Red, 50% Green, 100% Blue, 30% Opacity
```

Middle Tennessee State University

OpenGL Syntax Examples

- Unsigned byte – color component values range from 0 to 255 (same as C's unsigned char).

```
glColor3ub (0, 127, 255);  
This is: 0% Red, 50% Green, 100% Blue  
glColor4ub (0, 127, 255, 76);  
This is 0% Red, 50% Green, 100% Blue, 30% Opacity  
...
```

Middle Tennessee State University

Setting Drawing Colors in GL

- `glColor3f(red, green, blue);`
 - `glColor3f(1.0, 0.0, 0.0);` // red
 - `glColor3f(0.0, 1.0, 0.0);` // green
 - `glColor3f(0.0, 0.0, 1.0);` // blue
 - `glColor3f(0.0, 0.0, 0.0);` // black
 - `glColor3f(1.0, 1.0, 1.0);` // bright white
 - `glColor3f(1.0, 1.0, 0.0);` // bright yellow
 - `glColor3f(1.0, 0.0, 1.0);` // magenta
 - `glColor3f(0.0, 1.0, 1.0);` //cyan
- More colors described in the book

Middle Tennessee State University

Windowing with OpenGL

- OpenGL is independent of any specific window system
- OpenGL can be used with different window systems
 - X windows (GLX)
 - MFC (WGL)
 - ...
- GLUT provide a portable API for creating window and interacting with I/O devices

Middle Tennessee State University

GLUT

- Developed by Mark Kilgard
- Hides the complexities of differing window system APIs
 - Default user interface for class projects
- Glut routines have prefix **glut**
 - **glutCreateWindow()** ...
- Has very limited GUI interface
- **GLUI** is the C++ extension of glut that provides buttons, checkboxes, radio buttons, etc.

Middle Tennessee State University

Glut Routines

- **Initialization:**
glutInit() processes (and removes) command-line arguments that may be of interest to glut and the window system and does general initialization of Glut and OpenGL.
 - Must be called before any other glut routines
- **Display Mode:**
The next procedure, **glutInitDisplayMode()**, performs initializations informing OpenGL how to set up the frame buffer.
 - Display Mode Meaning
 - **GLUT_RGB** Use RGB colors
 - **GLUT_RGBA** Use RGB plus alpha (for transparency)
 - **GLUT_INDEX** Use indexed colors (not recommended)
 - **GLUT_DOUBLE** Use double buffering (recommended)
 - **GLUT_SINGLE** Use single buffering (not recommended)
 - **GLUT_DEPTH** Use depth-buffer (for hidden surface removal.)

Middle Tennessee State University

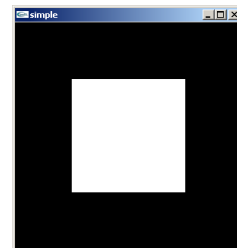
Glut Routines

- Window Setup
 - `glutInitWindowSize(int width, int height)`
 - `glutInitWindowPosition(int x, int y)`
 - `glutCreateWindow(char* title)`

Middle Tennessee State University

A Simple Program

Generate a square on a solid background



Middle Tennessee State University

cube.cpp

```
if using Windows, include the following
#include <Windows.h>
#include <gl/GL.h>
#include <gl/GLU.h>
#include <gl/glut.h>

if using linux, include the following
#include <GL/glut.h>

compile with:
gcc program.cpp -o RunProgram -I/usr/X11R6/include/ -L/usr/
X11R6/lib -lX11 -lXi -lglut -lGL -lGLU

if using Mac OS X, include these:
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>
```

Middle Tennessee State University

cube.cpp

```
int main(int argc, char** argv)
{
    glutInit(&argc,argv)
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(Width,Height);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Display Cube");
    glutDisplayFunc(Draw);

    MyInit();
    glutMainLoop();

    return 0;
}
```

Middle Tennessee State University

cube.cpp

```
void Draw()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();

    glFlush();
}
```

Middle Tennessee State University

Closer Look at the main()

```
int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(Width,Height);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Display Cube");
    glutDisplayFunc(draw);

    MyInit();
    glutMainLoop();
    return 0;
}
```

define window properties

display callback

set OpenGL state

enter event loop

Middle Tennessee State University

MyInit()

```
void MyInit()
{
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glColor3f(1.0, 1.0, 1.0);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
}
```

Annotations:

- black clear color (points to 0.0, 0.0, 0.0)
- opaque window (points to 1.0)
- fill/draw with white (points to 1.0, 1.0, 1.0)
- Define clipping window (points to gluOrtho2D)

Middle Tennessee State University

Callbacks

- Virtually all interactive graphics programs are event driven
- Glut uses callbacks to handle events
 - Windows system invokes a particular procedure when an event of particular type occurs.
 - MOST IMPORTANT: display event
 - Signaled when window first displays and whenever portions of the window reveals from blocking window
 - **glutDisplayFunc(void (*func)(void))** registers the display callback function
- Running the program: **glutMainLoop()**
 - Main event loop. Never exit()

Middle Tennessee State University

Basic Drawing in OpenGL

- We have learned how to create a window
- Simple 2D drawing
 - No lighting and shading
- OpenGL coordinate system has different origin from the window system
 - Uses lower left corner instead of upper left corner as origin

Middle Tennessee State University

OpenGL Drawing

- Steps in the display function
 1. Clear the window
 2. Set drawing attributes
 3. Send drawing commands
 4. Flush the buffer

Middle Tennessee State University

Step 1: Clear the Window

- **glClear(GL_COLOR_BUFFER_BIT)**
 - clears the frame buffer by overwriting it with the background color.
 - Background color is a **state** set by **glClearColor(GLfloat r, GLfloat g, GLfloat b, GLfloat a)** in **MyInit()**.
- **void glClear(GLbitfield mask)**
 - **Four masks:**
 - GL_COLOR_BUFFER_BIT
 - GL_DEPTH_BUFFER_BIT
 - GL_ACCUM_BUFFER_BIT
 - GL_STENCIL_BUFFER_BIT

Middle Tennessee State University

Step 2: Drawing Attributes: Color

- **glColor3f(GLfloat r, GLfloat g, GLfloat b)** sets the drawing color
 - **glColor3d()**, **glColor3ui()** can also be used
 - Remember OpenGL is a state machine
 - Once set, the attribute applies to all subsequent defined objects until it is set to some other value
 - **glColor3fv()** takes a flat array as input

Middle Tennessee State University

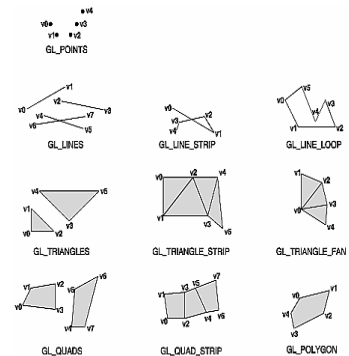
Step 2: Drawing Attributes

- Besides **glVertex()** commands, other attributes commands can also be used between **glBegin()** and **glEnd()**, e.g. **glColor3f()**.
- There are more drawing attributes than color
 - Point size: **glPointSize()**
 - Line width: **glLineWidth()**
 - Dash or dotted line: **glLineStipple()**
 - Polygon pattern: **glPolygonStipple()**
 - ...

Middle Tennessee State University

Step 3: Drawing Commands

- Simple Objects
glRectf()
- Complex Objects
 - Use construct **glBegin(mode)** and **glEnd()** and a list of vertices in between
 - **glBegin(mode);**
 glVertex(v0);
 glVertex(v1);
 ...
 glEnd();



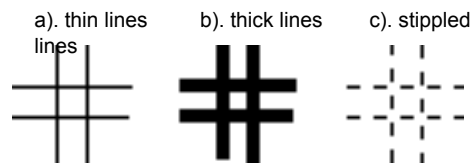
Drawing Lines

- `glBegin (GL_LINES); //draws one line`
 - `glVertex2i (40, 100); // between 2 vertices`
 - `glVertex2i (202, 96);`
- `glEnd ();`
- `glFlush();`
- If more than two vertices are specified between `glBegin(GL_LINES)` and `glEnd()` they are taken in pairs, and a separate line is drawn between each pair.

Middle Tennessee State University

Line Attributes

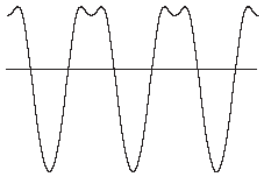
- Color, thickness, stippling.
- `glColor3f();` sets color.
- `glLineWidth(4.0);` sets thickness. The default thickness is 1.0.
- `glLineStipple(2, 0x777);`



Middle Tennessee State University

Graphing

- Drawing line graphs: connect each pair of $(x, f(x))$ values
- How would you design a program to accomplish this?



Middle Tennessee State University

Step 2: Drawing Attributes

`glLineStipple()` demo: `stipple0.cpp`



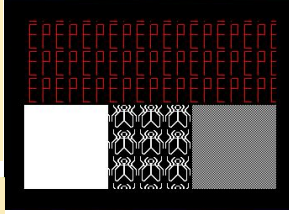
Middle Tennessee State University

Step 2: Drawing Attributes

```
GLubyte fly[] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x03, 0x80, 0x01, 0xc0, 0x06, 0xc0, 0x03, 0x60,
    0x04, 0x60, 0x06, 0x20, 0x04, 0x20, 0xc0, 0x20,
    0x04, 0x18, 0x18, 0x20, 0x04, 0xc0, 0x30, 0x20,
    0x04, 0x06, 0x60, 0x20, 0x44, 0x03, 0xc0, 0x22,
    0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
    0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
    0x66, 0x01, 0x80, 0x66, 0x33, 0x01, 0x80, 0xc0,
    0x19, 0x81, 0x98, 0xc0, 0xc1, 0x83, 0x30,
    0x07, 0xe1, 0x97, 0x80, 0x03, 0x34, 0xc0, 0x00,
    0x03, 0x31, 0x8c, 0xc0, 0x03, 0x33, 0xc0, 0xc0,
    0x06, 0x64, 0x26, 0x60, 0xc0, 0xc0, 0x33, 0x30,
    0x18, 0xc0, 0x33, 0x18, 0x10, 0xc0, 0x23, 0x08,
    0x10, 0x63, 0xc6, 0x08, 0x10, 0x30, 0xc0, 0x08,
    0x10, 0x16, 0x16, 0x08, 0x10, 0x00, 0x00, 0x08};

glRectf(25.0, 25.0, 125.0, 125.0);
glEnable(GL_POLYGON_STIPPLE);
glPolygonStipple(fly);
glRectf(125.0, 25.0, 225.0, 125.0);

glPolygonStipple()
```



Middle Tennessee State University

Polygon Issues

- OpenGL will only display polygons correctly that are
 - Simple**: edges cannot cross
 - Convex**: All points on line segment between two points in a polygon are also in the polygon
 - Flat**: all vertices are in the same plane
- User program can check if above true
 - OpenGL will produce output if these conditions are violated but it may not be what is desired
- Triangles satisfy all conditions



Middle Tennessee State University

Polygon Issues

- How to draw a circle?
 - circle_list demo
- How to draw this?
 - Polygon symbol
 - polygonSymbol demo

```
//glBegin(GL_POLYGON); // for solid circle
glBegin(GL_LINE_STRIP); // for unfilled circle
// Generate the points of the circle
for( int i=0; i<=numPoints; i++ )
{
    angle = i * (2.0*PI/numPoints);
    x = cos( angle )*radius;
    y = sin( angle )*radius;
    glVertex2f( x, y );
}
glEnd();
```



Middle Tennessee State University

Simple User Interaction with Mouse and Keyboard

- Register functions:
 - `glutMouseFunc (myMouse);`
 - `glutKeyboardFunc (myKeyboard);`
- Write the function(s)
- NOTE** that any drawing you do when you use these functions must be done **IN** the mouse or keyboard function, OR in a function called from within mouse or keyboard callback functions.

Middle Tennessee State University

Example Mouse Function

- `void myMouse(int button, int state, int x, int y);`
- Button is one of `GLUT_LEFT_BUTTON`, `GLUT_MIDDLE_BUTTON`, or `GLUT_RIGHT_BUTTON`.
- State is `GLUT_UP` or `GLUT_DOWN`.
- X and y are mouse position at the time of the event.

Middle Tennessee State University

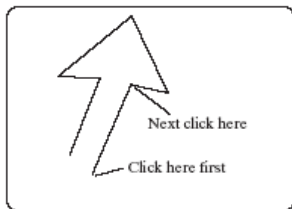
Example Mouse Function

- The x value is the number of pixels from the left of the window.
- The y value is the number of pixels *down* from the top of the window.
- In order to see the effects of some activity of the mouse or keyboard, the mouse or keyboard handler *must* call either `myDisplay()` or `glutPostRedisplay()`.

Middle Tennessee State University

Polyline Control with Mouse

- Example use:



Left click to draw
Right click to stop/clear

Middle Tennessee State University

Code for Mouse-controlled Polyline

```
void myMouse(int button, int state, int x, int y)
{
    //define NUM 20
    static GLintPoint List[NUM];
    static int last = -1; // last index used so far

    // test for mouse button as well as for a full array
    if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN && last < NUM - 1)
    {
        List[++last].x = x; // add new point to list
        List[last].y = screenHeight - y; // window height is 480
        glClear(GL_COLOR_BUFFER_BIT); // clear the screen
        glBegin(GL_LINE_STRIP); // redraw the polyline
            for(int i = 0; i <= last; i++)
                glVertex2i(List[i].x, List[i].y);
        glEnd();
        glFlush();
    }
    else if(button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
        last = -1; // reset the list to empty
}
```

Middle Tennessee State University

Using Mouse Motion Functions

- `glutMotionFunc(myMovedMouse);`
 - // moved with button held down
- `glutPassiveMotionFunc(myMovedMouse);`
 - // moved with buttons up
- `myMovedMouse(int x, int y);`
 - x and y are the position of the mouse when the event occurred.
- Code for drawing rubber rectangles using these functions is in Fig. 2.41.

Middle Tennessee State University

Example Keyboard Function

- Parameters to the function will always be `(unsigned char key, int mouseX, int mouseY)`.
- The y coordinate needs to be flipped by subtracting it from `screenHeight`.
- Body is a switch with cases to handle active keys (key value is ASCII code).
- Remember to end each case with a `break`!

Middle Tennessee State University

Example Keyboard Function

```
void myKeyboard(unsigned char theKey, int mouseX, int mouseY)
{
    GLint x = mouseX;
    GLint y = screenHeight - mouseY; // flip y value

    switch(theKey)
    {
        case 'p': drawDot(x, y);
                  break; // draw dot at mouse
        case 'E': exit(-1); // terminate the program
        default: break; // do nothing
    }
}
```

Middle Tennessee State University

Using Menus

- Both GLUT and GLUI make menus available.
- GLUT menus are simple, and GLUI menus are more powerful.
- Menus can be used to allow users to select options during the execution of your program

Middle Tennessee State University

GLUT Menu Callback Function

- `int glutCreateMenu(myMenu);` //returns menu ID
- `void myMenu(int num);` //handles choice num
- `void glutAddMenuEntry(char* name, int value);` // value used in myMenu switch to handle choice
- `void glutAttachMenu(int button);`

// one of GLUT_RIGHT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_LEFT_BUTTON
 – Usually GLUT_RIGHT_BUTTON

Middle Tennessee State University

GLUT subMenus

- Create a subMenu first, using menu commands, then add it to main menu.
 - A submenu pops up when a main menu item is selected.
- `glutAddSubMenu (char* name, int menuID);`
- menuID is the value returned by glutCreateMenu when the submenu was created

```
int submenu1, submenu2;

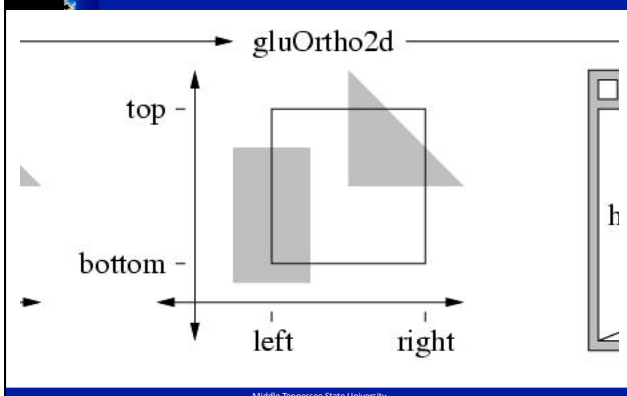
submenu1 = glutCreateMenu(selectMessage);
glutAddMenuEntry("abc", 1);
glutAddMenuEntry("ABC", 2);

submenu2 = glutCreateMenu(selectColor);
glutAddMenuEntry("Green", 1);
glutAddMenuEntry("Red", 2);
glutAddMenuEntry("White", 3);

glutCreateMenu(selectFont);
glutAddMenuEntry("9 by 15", 0);
glutAddMenuEntry("Times Roman 10", 1);
glutAddMenuEntry("Times Roman 24", 2);
glutAddSubMenu("Messages", submenu1);
glutAddSubMenu("Color", submenu2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

Middle Tennessee State University

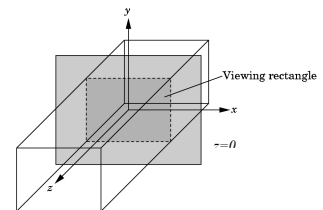
Projection and Viewport



Middle Tennessee State University

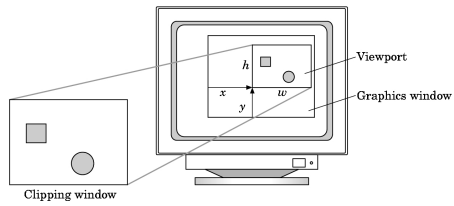
Orthographic projection

- Orthographic projection used for 2D drawing, Perspective project often used for 3D drawing
- 2D Viewing: Orthographic View
 - `gluOrtho2D(left, right, bottom, top)`
 - Specifies the coordinates of 2D region to be projected into the viewport.
 - Any drawing outside the region will be automatically clipped away.



Viewports

- Do not have to use the entire window for the image: `glViewport(x, y, w, h)`
- Values in pixels (screen coordinates)



Middle Tennessee State University