Rectangle Man

Lecture 9

Robb T. Koether

Hampden-Sydney College

Fri, Sep 8, 2017

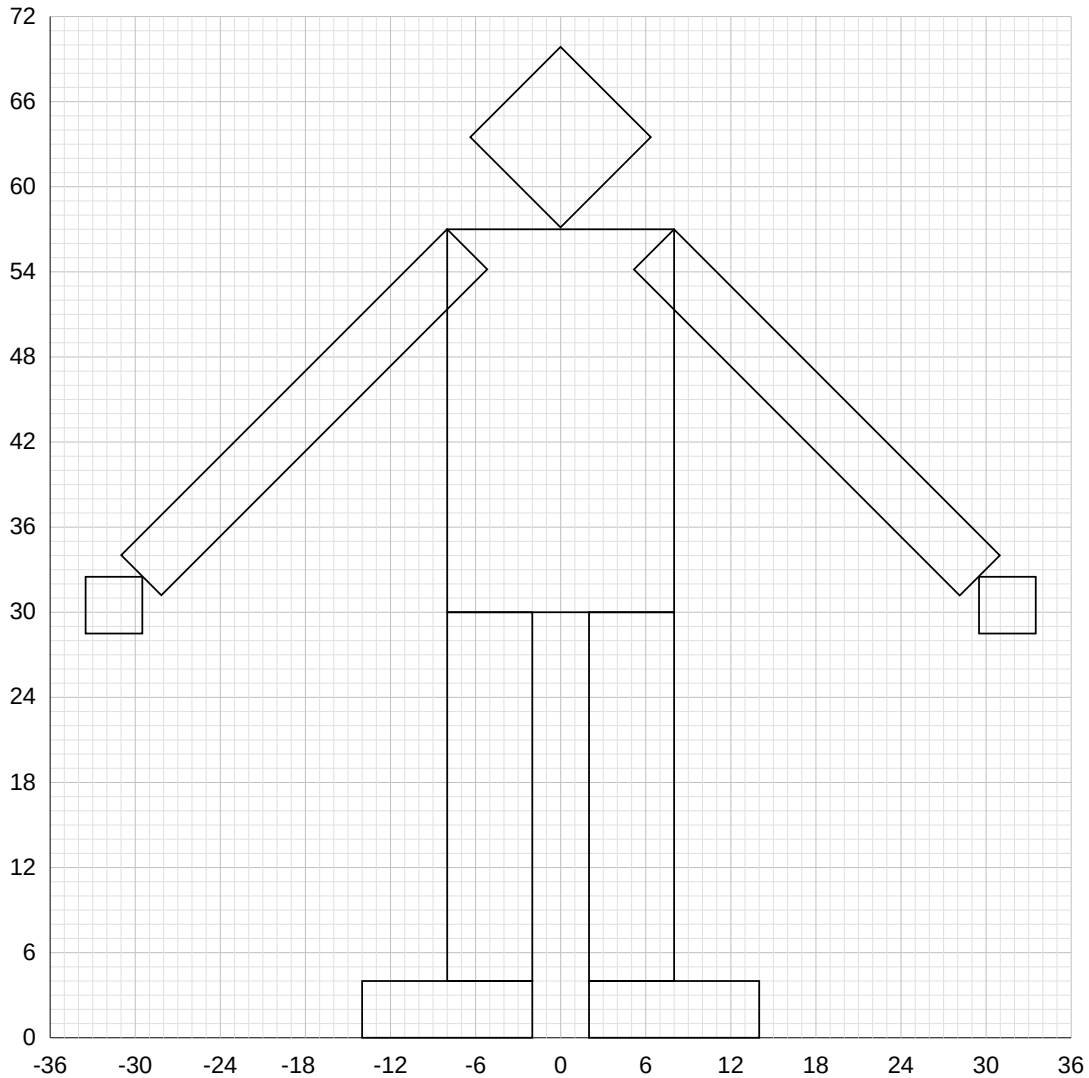
Outline

Outline

Drawing “Rectangle Man”

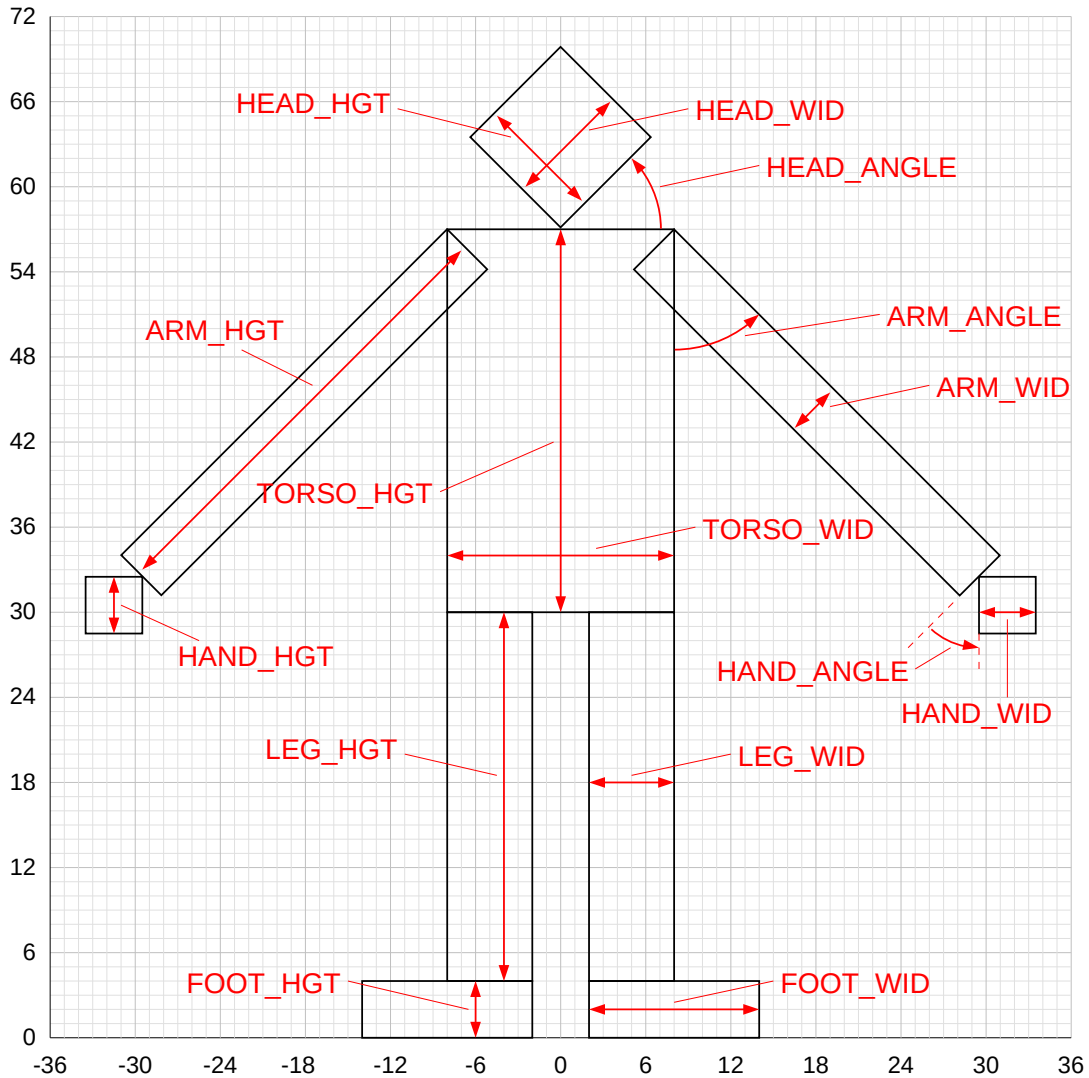
- We will draw an object (Rectangle Man) that is created entirely from our basic rectangle (unit square), using transformations.

Drawing “Rectangle Man”



Rectangle Man

Drawing “Rectangle Man”



Rectangle Man's Dimensions

The “Rectangle Man” Parameters

The “Rectangle Man” Parameters

```
//  The foot
    const float FOOT_WID = 12.0f;
    const float FOOT_HGT = 4.0f;
//  The leg
    const float LEG_WID = 6.0f;
    const float LEG_HGT = 22.0f;
    const float LEG_GAP = 4.0f;
        ⋮
```

- Assign a symbolic name to every parameter.

The “Rectangle Man” Parameters

The “Rectangle Man” Parameters

```
//  The foot
    const float FOOT_WID = 12.0f;
    const float FOOT_HGT = 4.0f;
//  The leg
    const float LEG_WID = 6.0f;
    const float LEG_HGT = 22.0f;
    const float LEG_GAP = 4.0f;
        ⋮
```

- Assign a symbolic name to every parameter.
- Don’t argue. Just do it.

Outline

The Modelview Stack

- Pushing and popping are used to “remember” previous transformations.
- The basic pattern is
  - Push the current matrix onto the stack (to remember it).
  - Perform a series of geometric transformations and draw an object.
  - Pop the current matrix off the stack, thereby restoring the former “current matrix.”

Manipulating the Stack

Initialize the stack

```
ModelStack model_stack(...);          // Global
model_stack.setModelLoc(model_loc); // In init();
model_stack.init();                    // In display()
```

- Declare the stack globally.
- After defining the variable `model_loc` in the `init()` function, store its value in the `ModelStack` object.
- Initialize it in the `display()` function before any drawing is done.

Manipulating the Stack

Drawing an Object

```
model_stack.push();                                          // Push CM
{
    model_stack.mult(translate(2.0f, 0.0f, 0.0f));      // Matrix T
    model_stack.mult(rotate(90.0f, 0.0f, 0.0f, 1.0f)); // Matrix R
    model_stack.mult(scale(1.0f, 4.0f, 1.0f));          // Matrix S
    model_stack.toShader();
    drawRectangle();
}
model_stack.pop();                                           // Pop CM
```

- Transform and draw an object without losing the previous transformation.
- The effect is $\mathbf{v}' = \mathbf{CM} * \mathbf{T} * \mathbf{R} * \mathbf{S} * \mathbf{v}$, where **CM** is the current model matrix.

Outline

The `display()` Function

The `display()` Function

```
void display()
{
        :
    model_stack.init();
    model_stack.mult(translate(offset_x, offset_y, 0.0f));
    model_stack.mult(translate(x_0, y_0, 0.0f));
    model_stack.mult(rotate(rot_angle, 0.0f, 0.0f, 1.0f));
    model_stack.mult(scale(scale_factor, scale_factor, 1.0f));
    model_stack.mult(translate(-x_0, -y_0, 0.0f));
    model_stack.toShader();
        :
    drawRectangleMan();  // Draw Rect Man in model coords
}
```

The `drawRectangleMan()` Function

The `drawRectangleMan()` Function

```
void drawRectangleMan()
{
    model_stack.toShader();
    drawHalfMan();                  // Draw right half
    model_stack.push();
    {
        model_stack.mult(scale(-1.0f, 1.0f, 1.0f));
        drawHalfMan();              // Draw left half
    }
    model_stack.pop();
    drawHead();
}
```

The `drawFoot()` Function

The `drawFoot()` Function

```
void drawFoot()
{
    model_stack.push();
    {
        model_stack.mult(translate(LEG_GAP/2.0f, 0.0f, 0.0f));
        model_stack.mult(scale(FOOT_WID, FOOT_HGT, 1.0f));
        model_stack.toShader();
        drawRectangle();    // Draw basic square
    }
    model_stack.pop();
}
```

The `drawArm()` Function

The `drawArm()` Function

```
void drawArm()
{
    model_stack.push();
    {
        model_stack.mult(translate(TORSO_WID/2.0f,
            LEG_HGT + TORSO_HGT, 0.0f));
        model_stack.mult(rotate(ARM_ANGLE, 0.0f, 0.0f, 1.0f));
        model_stack.mult(translate(-ARM_WID, -ARM_HGT, 0.0f));
        model_stack.push();
        {
            model_stack.mult(scale(ARM_WID, ARM_HGT, 1.0f));
            model_stack.toShader();
            drawRectangle();
        }
        model_stack.pop();
        drawHand();
     }
    model_stack.pop();
}
```

The `drawHand()` Function

The `drawHand()` Function

```
void drawHand()
{
    model_stack.push();
    {
        model_stack.mult(translate(ARM_WID/2.0f, 0.0f, 0.0f));
        model_stack.mult(rotate(-HAND_ANGLE, 0.0f, 0.0f, 1.0f));
        model_stack.mult(translate(0.0f, -HAND_HGT, 0.0f));
        model_stack.mult(scale(HAND_WID, HAND_HGT, 1.0f));
        model_stack.toShader();
        drawRectangle();
    }
    mode_stack.pop();
}
```

Outline

Assignment

Assignment

- No new assignment. Keep working on Assignment 8.