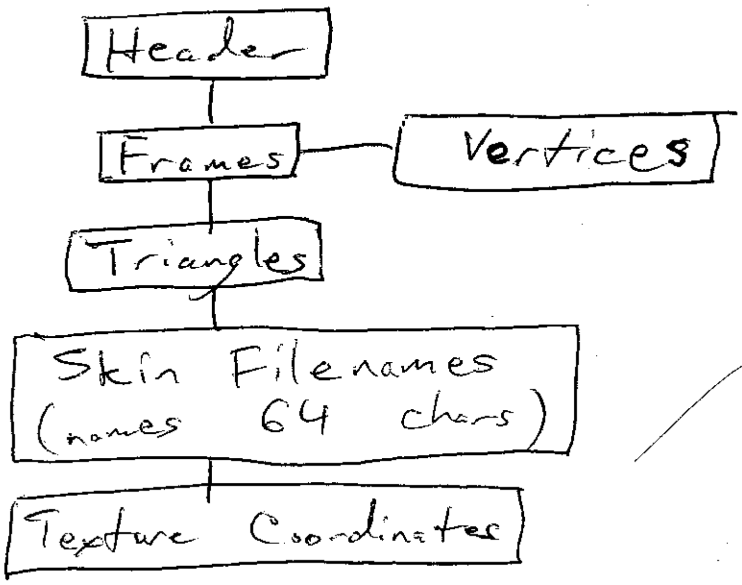
1)



2)

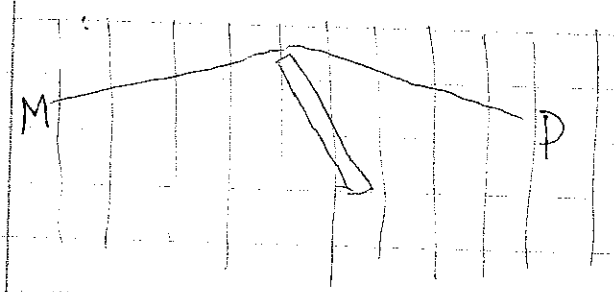
a) create your GDI drawing tool (CPen, CBrush, etc.)

b) declare a pointer to a GDI drawing tool (CPen, CBrush, etc.) named pOld

c) use pDC→SelectObject to use your created drawing tool. Set the pOld pointer to the returned value.

d) When done drawing, use pDC→SelectObject to choose pOld as your tool.

3)



→ keep a priority queue of nodes to search from, starting with only Monster's position

– grab node from queue, if player output the path, if already visited ~~&~~ pull off another node (would have a list of visited nodes)

– for each ~~[illegible]~~ neighbor node ~~[illegible]~~ ~~[illegible]~~ add ~~&~~ the ~~node~~ neighbor to the priority queue using the cost to get to this ~~node~~ neighbor plus the estimated cost ~~&~~ to go from this ~~node~~ neighbor to the player.

repeat until reach player

4)

```
PIXELFORMATDESCRIPTOR pixelFormat;
int pixelFormat_index;
HGLRC openglrenderingcontext
pixelformatindex = ChoosePixelFormat(hdc, &pixelformat);
SelectPixelFormat(hdc, pixelformatindex, &pixelformat);
openglrenderingcontext = wglCreateContext(hdc_view);
wglMakeCurrent(hdc_view, openglrendering context);
glBegin(GL_POINTS);
    glVertex2f(0.0, 0.0);
    glVertex2f(0.0, 1.0);
    glVertex2f(1.0, 1.0);
    glVertex2f(1.0, 0.0);
glEnd();
glFinish();
SwapBuffers(hdc_view);
```

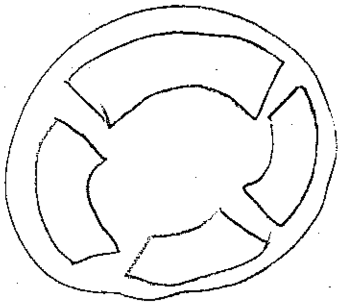
Override World Shape

5.) Virtual int World Shape () { ~~return cGame::SHAPE~~
    return cGame :: SHAPE_X SCROLLER; }

(6) Create an array of cCritterShowoneChild as buttons, which one child is for turned off state, one child is for turned on state.

Change critters to turned on state randomely, store the indexes of the critter in the order that they were turned on.

Use cListenerCursor to know which critter is clicked, compare with the array of indexes to check whether the order is correct.



Joshua Quick

Final Practice

1) Donkey Kong

Forces Needed:

cForceGravity - Applied to mario to keep him on floor & for jumping.
- applied to barrels too.

cForceObjSeek - For fire guys to chase down mario.

Listener needed:

cListenerHopper - allows you to run & jump.

Critters needed:

- cCritterWalls
- cCritterPropHammer
- cCritterPropDonkeyKong
- cCritterPropGirlFriend
- cCritterMario ← The player!
- cCritterLadder
- cCritterBarrel
- cCritterFlamingBarrel

8. Color sniffing is a way to determine the color off certain pixels.
The pop framework has the method,

```
COLORREF cCritter::sniff(const cVector &sniffLocation,
                         CPopView *pactiveView)
```

This tells you the color of the onscreen view corresponding to the given location.

In a race car game, we make the track one color and the walls and other off-road objects a different color.

Then, we can make the car stay on the track by doing the following:

In the critter's update code, you can have it look ahead and sniff to see if it is about to move off the track.
If it is about to move off the track, we can either
1. look for a better direction to move it.
2. make the car bounce off the wall.

9. A move box limits the movement of a critter. A ~~[illegible]~~ critter is unable to move beyond the bounds of their move box.

A drag box is used to limit where a critter can be dragged. A critter can be dragged beyond their movebox if the drag box bounds are different.

10. Our group followed the extreme programming paradigm.
~~only~~
~~We [illegible] altered the [illegible] [illegible] except~~
We only altered our project during our meetings.