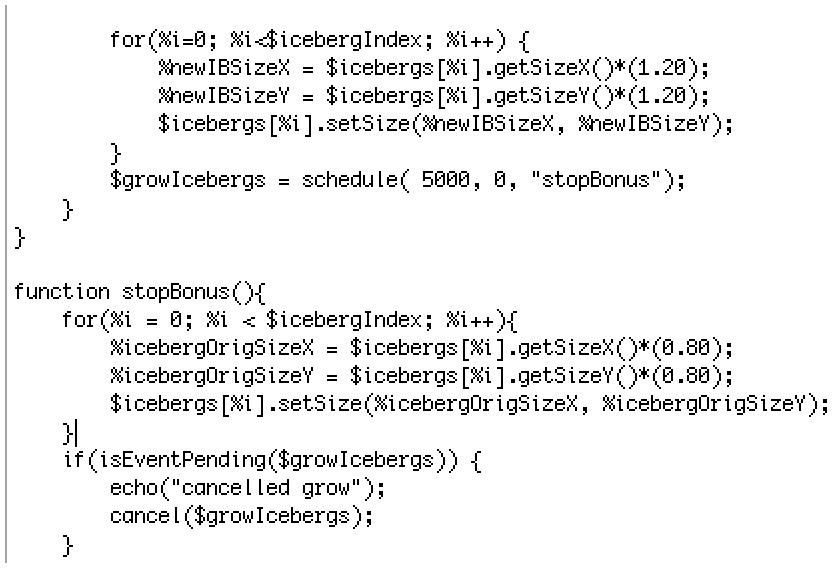
Dan's Portfolio 269-369

The bulk of my course after the obvious introduction to Torque Game Builder and GIMP the first week was spent learning Torque Script. I decided rather than spend time doing tutorials which had aspects I may not use or understand, I took the approach that I would simply try to do what I wanted with the game until I couldn't and then look for reference in the documentation or the tutorials for similar situations. During the course Kevin and I were the main gameplay programmers for our group. We worked very closely on a number of the gamescript files, some more similar that others.

Alphabetically I will start from the beginning...

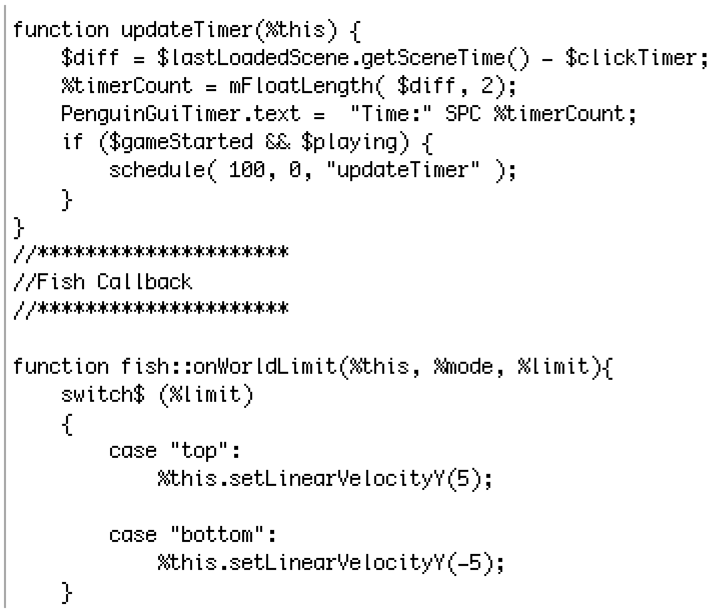
Fish.cs is the main code for the bonus fish that grows the icebergs for a certain amount of time. The collision code here is similar to the iceberg collision code, it looks at the object colliding with it and determines if it is the penguin, if so, the object is mounted and the game continues. The dilemma we came across with the bonus fish was how to grow just the icebergs that the penguin could reach. Then we realized that to limit the amount of icebergs grown would also limit the rate of the penguins progression and instead we should grow our entire set of icebergs via a simple iteration and an amplification of the size by 20%. Then scheduling another function which does the opposite.



```
        for(%i=0; %i<$icebergIndex; %i++) {
            %newIBSizeX = $icebergs[%i].getSizeX()*(1.20);
            %newIBSizeY = $icebergs[%i].getSizeY()*(1.20);
            $icebergs[%i].setSize(%newIBSizeX, %newIBSizeY);
        }
        $growIcebergs = schedule( 5000, 0, "stopBonus");
    }
}

function stopBonus(){
    for(%i = 0; %i < $icebergIndex; %i++){
        %icebergOrigSizeX = $icebergs[%i].getSizeX()*(0.80);
        %icebergOrigSizeY = $icebergs[%i].getSizeY()*(0.80);
        $icebergs[%i].setSize(%icebergOrigSizeX, %icebergOrigSizeY);
    }
    if(isEventPending($growIcebergs)) {
        echo("cancelled grow");
        cancel($growIcebergs);
    }
```

Game.cs is a very involved script, it does a lot of things in our game. It handles most of the gui changes and updates. Likewise it sets many important global variables such as $playing, and $gameStarted. Finally it handles all input and draws all of our animated and static sprites. My two serious contributions to this script were involved in the setting fish world limit motions, and adjusting the timer code.

The next file I had a hand is, is arguably the most crucial to our project, the penguin. After all he's the source of all the fun 😀 The most important role I had in penguin.cs is the follow mouse behavior, it took quite some time with the behavior tutorial before I realized how to implement one in script, and edit the script for the behavior itself. In retrospect it was relatively easy. The behavior takes in two fields, updates them constantly, calculates the position of the mouse, the difference between the object and the mouse position and then adjusts based on trackingspeed inputted. This allowed us to use the penguin as the primary game controller.



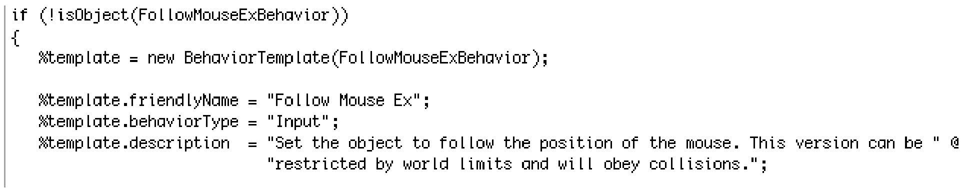
```
function updateTimer(%this) {
    $diff = $lastLoadedScene.getSceneTime() - $clickTimer;
    %timerCount = mFloatLength( $diff, 2);
    PenguinGuiTimer.text =  "Time:" SPC %timerCount;
    if ($gameStarted && $playing) {
        schedule( 100, 0, "updateTimer" );
    }
}
//*********************
//Fish Callback
//*********************

function fish::onWorldLimit(%this, %mode, %limit){
    switch$ (%limit)
    {
        case "top":
            %this.setLinearVelocityY(5);

        case "bottom":
            %this.setLinearVelocityY(-5);
    }
```

FollowMouseExBehavior

Penguin.cs

The majority of my time beyond these fields was invested in debugging, and in compiling all the gui script, artwork, music, and game script into one cohesive program, believe it or not making them all play nicely with each other took the better part of 2 days. It was a great course and I look forward to next semester.

Dan



```
if (!isObject(FollowMouseExBehavior))
{
   %template = new BehaviorTemplate(FollowMouseExBehavior);

   %template.friendlyName = "Follow Mouse Ex";
   %template.behaviorType = "Input";
   %template.description  = "Set the object to follow the position of the mouse. This version can be " @
                            "restricted by world limits and will obey collisions.";
   %template.addBehaviorField(followX, "Follow the mouse's X position", bool, true);
   %template.addBehaviorField(followY, "Follow the mouse's Y position", bool, true);
   %template.addBehaviorField(trackingSpeed, "The rate at which the object will move toward the mouse", float, 6.0);
}

function FollowMouseExBehavior::onBehaviorAdd(%this)
{
   %this.owner.enableUpdateCallback();
}

function FollowMouseExBehavior::onUpdate(%this)
{
   if (!isObject(sceneWindow2d))
      return;

   %mousePos = sceneWindow2D.getMousePosition();
   %position = %this.owner.position;

   %difference = t2dVectorSub(%mousePos, %position);
   %amount = t2dVectorScale(%difference, %this.trackingSpeed);

   if (%this.followX)
      %this.owner.setLinearVelocityX(%amount.x);

   if (%this.followY)
      %this.owner.setLinearVelocityY(%amount.y);
}
```

```
    //set collision properties
    $penguin.setCollisionActive(true, true);
    //$penguin.setWorldLimit("BOUNCE", "-50 -35 1000 35");

    //set to follow mouse
    %mouseBehavior = FollowMouseExBehavior.createInstance();
    $penguin.addBehavior(%mouseBehavior);
    %mouseBehavior.trackSpeed = "3";

    //add to scene graph
    %sceneGraph.addToScene($penguin);

    $penguin.setFrameChangeCallback(true);
}
```