Before the lecture begins …

- Make sure you have Homework 2 turned in.
- Please update your framework at http://pclx.com/itcc for a version that *will* work on your compiler. Dev-C++ (g++) is now supported as well thanks to one of your classmates.

Intro to C++

Lecture 4

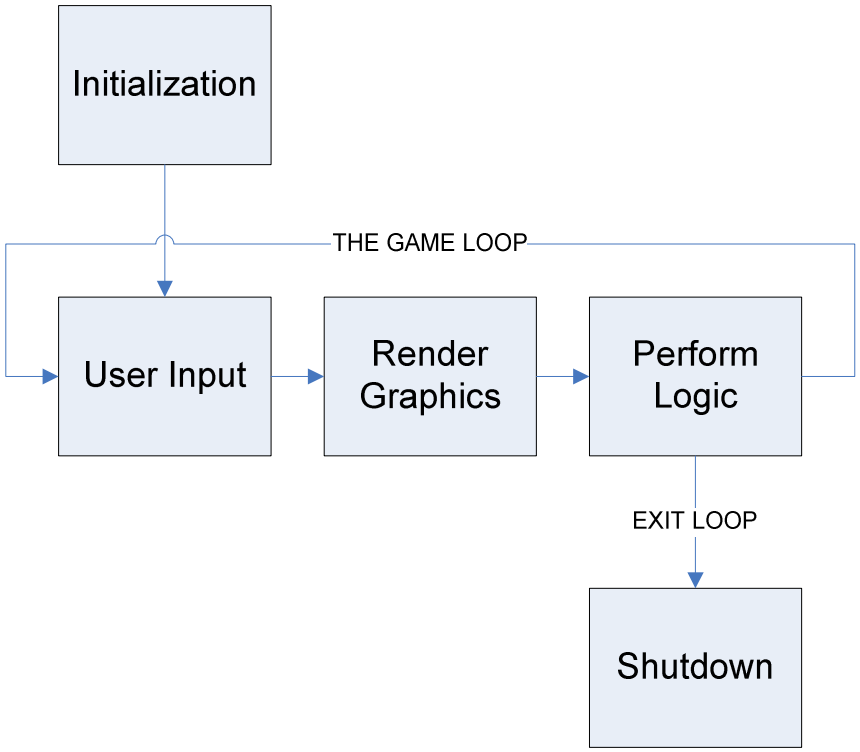
Pong review

How do I make a game?

- A computer game has many different parts to it, notably:
  - Initialization
  - Render Graphics
  - Game Logic
  - User Input
  - Shutdown

The Game Flowchart

- Initialization
- Render Graphics
- Game Logic
- User Input
- Shutdown



Pong: Where Do We Begin?

- Let's start out with rendering the graphics, since that's the fun part.
- So first, let's brainstorm what objects we need to draw:
  - Ball
  - Two paddles
  - Upper and lower screen boundaries

Pong: Screen Boundaries

- Let's begin with the easiest item to draw – the screen boundaries. Why is this easy? Because they won't move at all throughout the game.
- So I want two horizontal lines to appear on the screen, one at the top and one at the bottom. I'll make them dashed just for fun.

Pong: Screen Boundaries

```
// Draw the top/bottom screen boundaries as dotted lines
for (xIndex=0; xIndex<xRes; xIndex+=2)
{
        itcc->SetPixel(xIndex,      0, screenBoundColor);
        itcc->SetPixel(xIndex, yRes-1, screenBoundColor);
}
```

A few things to note:

- The variable names I've used are descriptive.
- The screen resolution is stored in yRes so I don't have to rewrite all of this code every time I change the resolution.
- The comment describes the function of the code block rather than give a literal description.
- 'i' isn't used as the counter variable, but something a little more descriptive ('xIndex') is used instead.

Pong: Screen Boundaries

- Now that I've written the code to display the boundaries, let's see what variables need to be declared and declare them:

```
int xRes = 640, yRes = 480;      // height (y) and width (x) of screen
int xIndex = 0, yIndex = 0;      // for loop indices

unsigned long screenBoundColor = RGB32(  0, 255, 255);  // cyan
```

Pong: Drawing the Ball

```
// Draw the ball
for (xIndex=0; xIndex<ballHeight; xIndex++)
{
        for (yIndex=0; yIndex<ballHeight; yIndex++)
        {
                itcc->SetPixel(xBall + xIndex, yBall + yIndex, ballColor);
        }
}
```

A few things to note:

- In general, you don't want to use constant numbers such as '640' in your code (it can get really confusing), so in the above everything is done symbolically.
- The above code will draw a ballHeightxballHeight ball on the screen with the upper left pixel at (xBall, yBall).
- We use two for loops to draw two dimensions of pixels

Pong: Drawing the Paddles

```
// Draw the left/right paddles
for (xIndex=0; xIndex<paddleWidth; xIndex++)
{
   for (yIndex=0; yIndex<paddleHeight; yIndex++)
   {
      itcc->SetPixel(xPaddleLeft  + xIndex, yPaddleLeft  + yIndex, paddleLeftColor );
      itcc->SetPixel(xPaddleRight + xIndex, yPaddleRight + yIndex, paddleRightColor);
   }
}
```

A few things to note:

- Each paddle has a common thickness and a height.
- The initial paddle positions are centered as follows:

```
int xPaddleLeft  = xPaddleFromEdge;
int yPaddleLeft  = yRes / 2 - paddleHeight / 2;
int xPaddleRight = xRes - paddleWidth - xPaddleFromEdge;
int yPaddleRight = yRes / 2 - paddleHeight / 2;
```

Pong: In-Bound Paddles

```
// ensure the paddles are in-bounds
if (yPaddleLeft  <= 0)    { yPaddleLeft  = 0;  }
if (yPaddleRight <= 0)    { yPaddleRight = 0;  }

if (yPaddleLeft  >= yRes-paddleHeight) { yPaddleLeft  = yRes - paddleHeight; }
if (yPaddleRight >= yRes-paddleHeight) { yPaddleRight = yRes - paddleHeight; }
```

A few things to note:

- If the paddle ever drifts off the screen, it is reset to one of the screen boundaries.

Pong: Moving the Ball

```
// calculate new ball coordinates
xBall += xBallVel;
yBall += yBallVel;
```

• Every frame, the ball's position is updated by adding on the ball's velocity in the x and y directions. Note that if xBallVel > 0, it will move to the right. If it = 0, the ball will be stationary. If it's < 0, the ball will move to the left.

Pong: Ball Bouncing

```
// Bounce off the top and bottom wall
if (yBall <= 0)
{
        yBallVel = ballSpeed;
}
else if (yBall >= yRes - ballHeight)
{
        yBallVel = -ballSpeed;
}
```

A few things to note:

- With the ball velocity scheme, note that the only two times we want to change the ball's y velocity are when the ball tries to leave the screen.
- Thus, if it tries to leave the top, we make the y velocity positive so the ball will go down. Vice versa for the bottom.

Pong: Paddle Bouncing

```
// bounce off the paddles
if (xBall < xPaddleLeft + paddleWidth - 1)
{
        // did it hit the paddle?
        if (    (yBall >= yPaddleLeft - ballHeight)
                 && (yBall <= yPaddleLeft + paddleHeight))
        {
                xBallVel = ballSpeed;
        }
        else
        {
                itcc->MsgBox("WINNER!", "Right player wins! Congrats!");
                isRunning = FALSE;
        }
}
```

Let's analyze this piece by piece.
The code for the right paddle bounce is similar.

Pong: Paddle Bouncing

```
if (xBall < xPaddleLeft + paddleWidth - 1)
```

• Here, we test to see whether the ball is one pixel to the left of the left paddle along the x axis.

```
if (      (yBall >= yPaddleLeft - ballHeight)
   && (yBall <= yPaddleLeft + paddleHeight))
```

• Here, we need to test to see whether the ball is inside a certain range of y values. Note that for the ball to be within the top and bottom of the paddle, it must be greater than the y coordinate of the top of the paddle and less than the y of the paddle's bottom.
• If this is NOT true, then the player missed the ball.

Pong: Input

```
// keys control the paddle position
if (KEYDOWN('A'))      {  yPaddleLeft--;  }
if (KEYDOWN('Z'))      {  yPaddleLeft++;  }
if (KEYDOWN(VK_UP))    {  yPaddleRight--; }
if (KEYDOWN(VK_DOWN)) {  yPaddleRight++; }

// escape key exists
if (KEYDOWN(VK_ESCAPE)) {  isRunning = FALSE;  }
```

A few things to note:

- We can use ‘A’-’Z’ in KEYDOWN to specify that a letter is pushed.
- The variable isRunning is used as one of the game loop conditions to continue execution in the loop. If this is ever FALSE, then the game loop will stop repeating.

Pong: Miscellaneous

```
Sleep (5);
```

- We tell the computer to “go to sleep” and not do anything for five milliseconds every frame. This causes the ball to slow down, but it also means that user input is sampled less frequently.

```
char textBuffer[50];
sprintf(textBuffer,"Player 1 Score: %d", scorePlayer1);
itcc->Text(textBuffer, x,y,color);
```

- This code fragment can be used to display a variable onto the screen. I’ll update the framework with a better C++ version sometime so that we can get in the habit of using C++ instead of this C routine sprintf.

TODO

- **EVERYONE**: PLEASE DOWNLOAD THE LATEST FRAMEWORK VERSION FROM THE SITE AND INSTALL IT. DO NOT USE OLDER FRAMEWORK VERSIONS. Instructions will be posted on installing the new framework after this lecture (they are the same as for the HW2 framework).
- Download ITCC_HW3.zip from the site (will not be available until Wednesday, July 14) http://www.pclx.com/itcc/, and complete the homework exercises, emailing them (FOR THIS WEEK ONLY) to itcc_teachers@pclx.com. Please do not resubmit solutions, even if they are revised. All homework must be submitted by 6:00am PST Monday, July 12.
- Some illustrative examples of the topics in this lecture are given in ITCC_HW3.zip.
- Look over the slides for the fifth lecture before Thursday, July 15.
- If you finish with the homework, experiment!