Images, Resources and Surfaces

Giovanni Dicanio
giovanni.dicanio@gmail.com

pluralsight
hardcore dev and IT training

Topics

- **Loading and displaying images**
  - Loading from files (including assets)
  - Loading from resources
- **Surface vs. Texture**
  - Accessing surface pixels
- **Demos**

Images: From File to Screen

Images: From File to Screen

- Create a gl::Texture object
- Load the image from file to the gl::Texture object
- Draw the gl::Texture to the screen using OpenGL

The gl::Texture Class

- Represents an OpenGL texture
- A block of bitmap data on the GPU
- Image on the graphics card

Photo by Advanced Micro Devices, Inc. (AMD)

http://bit.ly/1s4MXIH

Creating the Texture Object

- **Add a gl::Texture data member**
  - cinder/gl/Texture.h header

```
class DemoApp : public AppNative
{
public:
    void setup();
    void draw();

private:
    // This texture will contain the image loaded
    gl::Texture mTexture;
};
```

Loading the Image into Texture

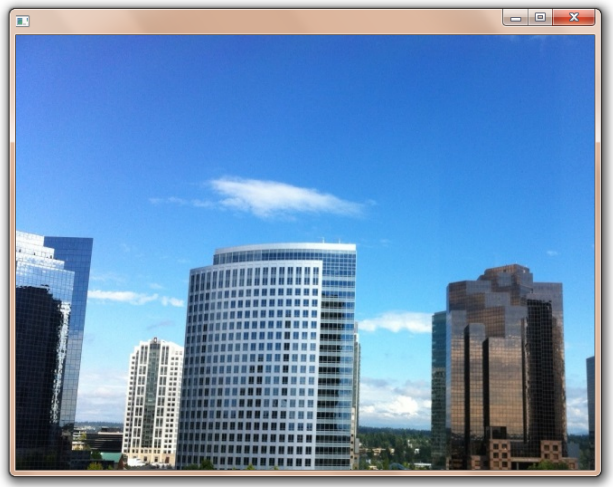
- **Call loadImage()**
  - cinder/ImageIo.h header

```
void DemoApp::setup()
{
    // Load the image into the gl::Texture object
    mTexture = loadImage("Bellevue.png");
}
```

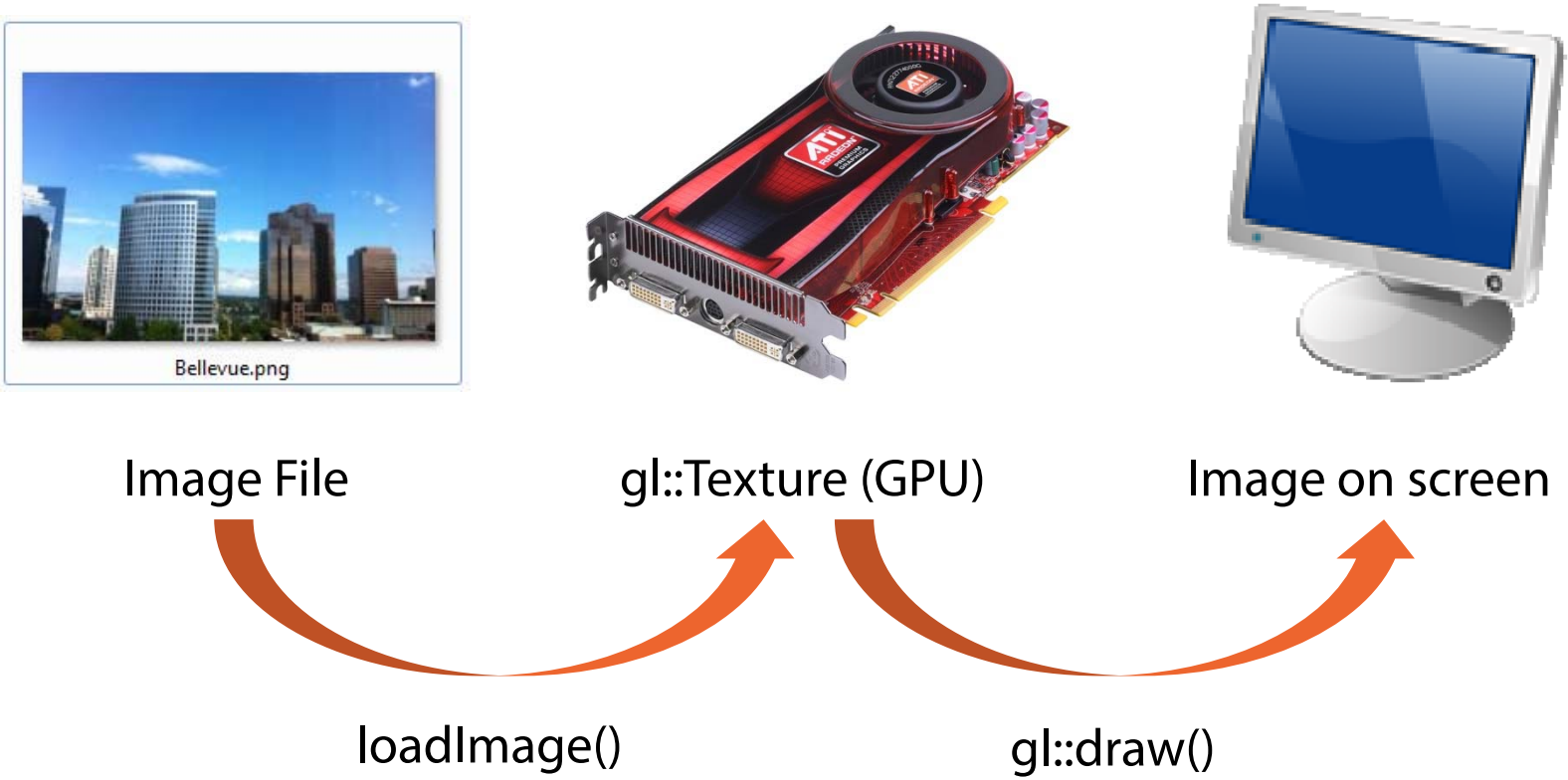
Drawing the Texture to the Screen

- Call gl::draw()

```
void DemoApp::draw()
{
    // Show the image
    gl::draw(mTexture);
}
```



Images: From File to Screen (Recap)



Cinder's Design: Division of Labor

- No gl::Texture::loadImage() *method*
- No gl::Texture::draw() *method*

```
mTexture.loadImage("Bellevue.png");

mTexture.draw();
```

Cinder's Design: Division of Labor

- **loadImage() is a *standalone* function**
- **gl::draw() is a *standalone* function**

```
void DemoApp::setup()
{
    // Load the image into the gl::Texture object
    mTexture = loadImage("Bellevue.png");
}
```

```
void DemoApp::draw()
{
    // Show the image
    gl::draw(mTexture);
}
```

Cinder's Design: Division of Labor

- Reuse the *same* image loading function for *different* classes

```
void DemoApp::setup()
{
    // Load the image into the gl::Texture object
    mTexture = loadImage("Bellevue.png");
}
```

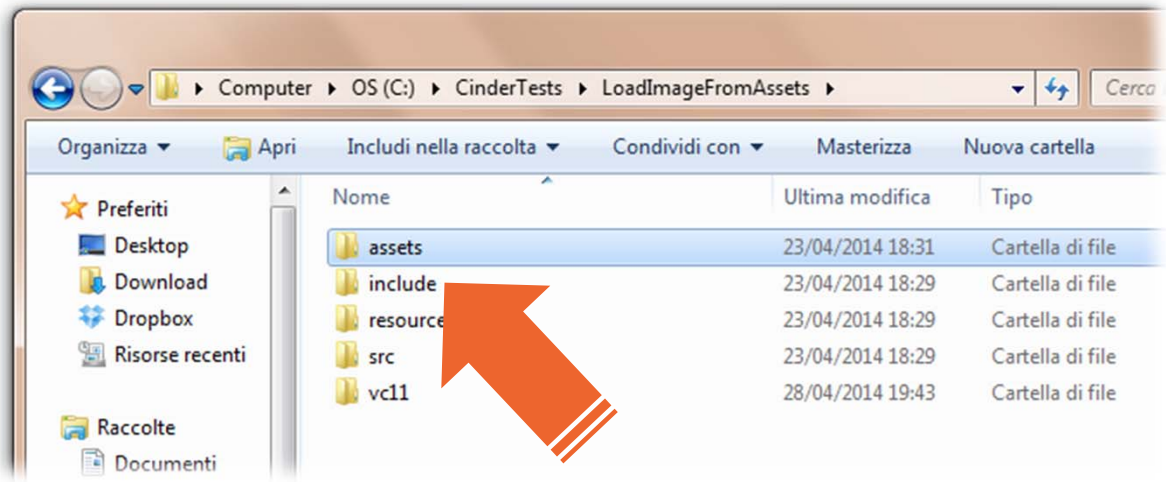
- ImageSourceRef loadImage(…)
- gl::Texture(ImageSourceRef …)
- Surface(ImageSourceRef …)

Assets

- Place images into the *assets* folder
- Call loadAsset()

Works for *any* file!

```
// Load the test image from assets
mTexture = loadImage(loadAsset("Bellevue.jpg"));
```



Demo: Loading Images From Assets

Assets: Pros and Cons

- **Assets Pros**
  - Very easy
- **Assets Cons**
  - Must redistribute the assets folder alongside the .EXE

assets folder

.EXE

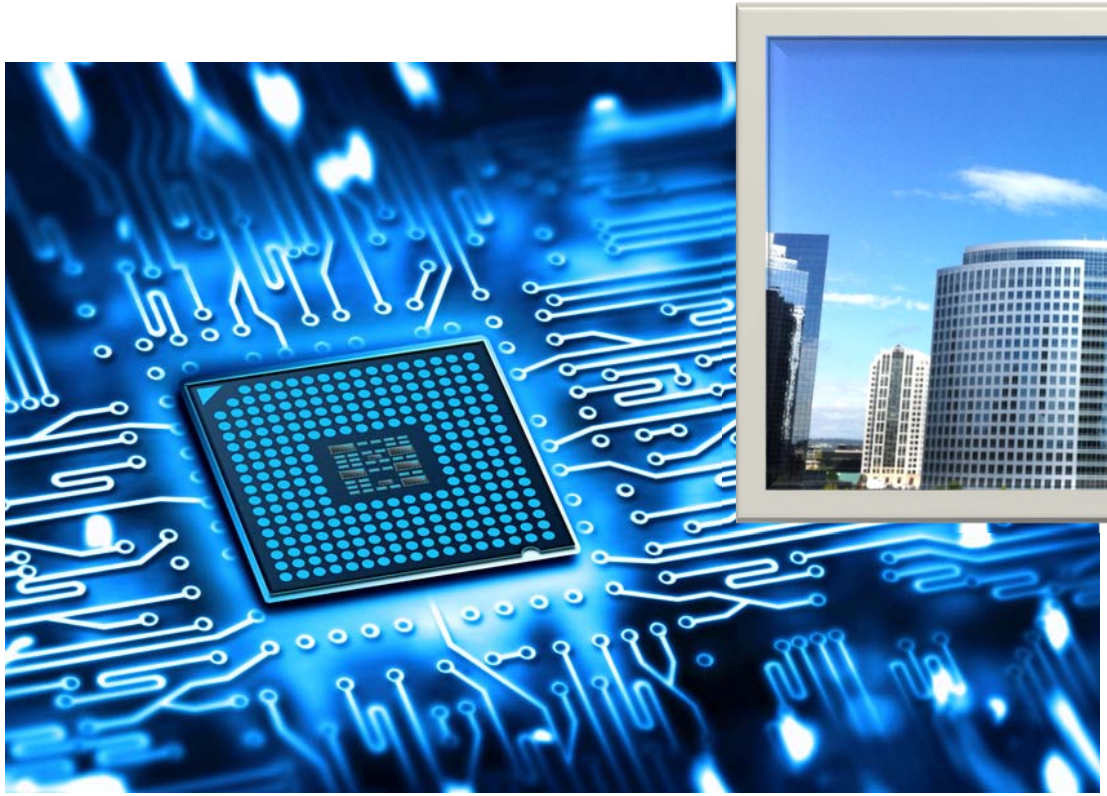
Resources

- Apps entirely self-contained
- Resources are baked into the .EXE

Demo: Loading Images From Resources

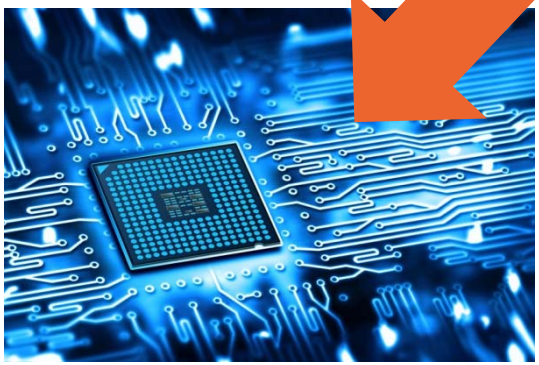
The Surface Class

Block of bitmap data on the CPU



Surface vs. gl::Texture

Block of bitmap data



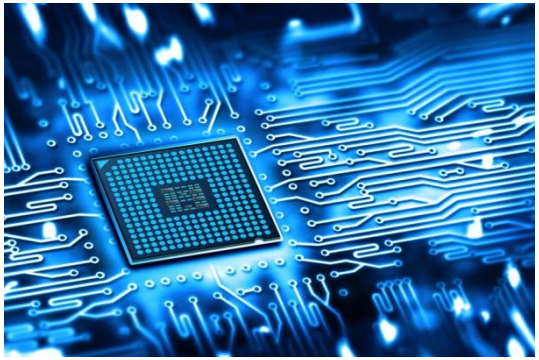
CPU → Surface

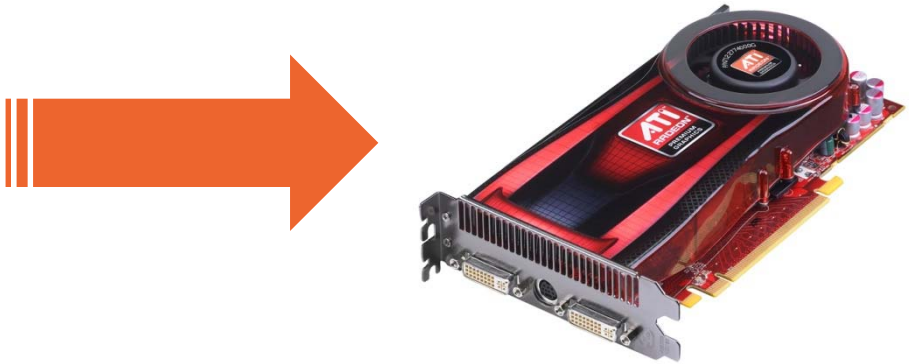
GPU → gl::Texture

Surface to Texture

```
Surface mySurface;
// Init mySurface with some bitmap data...

// Convert surface to texture
gl::Texture myTexture(mySurface);
```



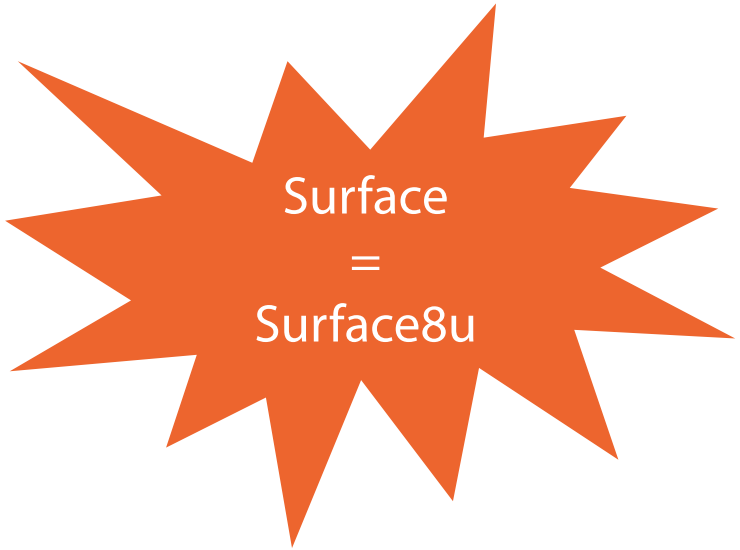


CPU → Surface

GPU → gl::Texture

Surface's «Flavors»

- **SurfaceT<T> class template**
- **T = uint8_t → Surface8u**
  - 8-bit unsigned integer version
  - R,G,B(,A) are uint8_t
  - Used the majority of the time
- **T = float → Surface32f**
  - 32-bit float
  - R,G,B(,A) are 32-bit float
  - Used for advanced image processing



Creating Surfaces

```
Surface mySurface;
```

Creates an empty surface

```
Surface mySurface( 800, 600, true );
```

Creates a surface of 800x600 (width x height) pixels with an alpha channel

```
Surface picture = loadImage("Bellevue.png");
```

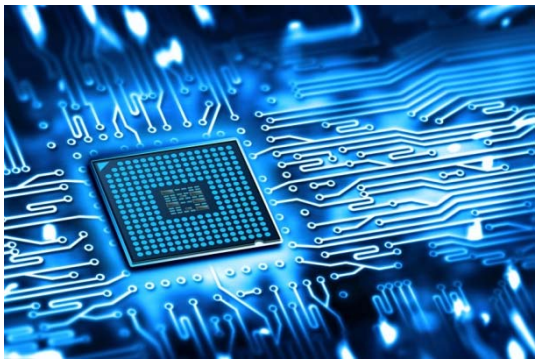
Creates a surface with the content of an image file

Surface From Texture

```
gl::Texture myTexture;
// ... bitmap data written somewhere else in the texture

Surface surfaceFromTexture(myTexture);
```

GPU → gl::Texture



CPU → Surface

Copying Surfaces

```
Surface surface1;
```

```
surface2
```

```
Surface surface2 = surface1;
```

Shallow Copy

Copying Surfaces (Deep Copy)

```
Surface surface1;
```

```
Surface surface2 = surface1.clone();
```

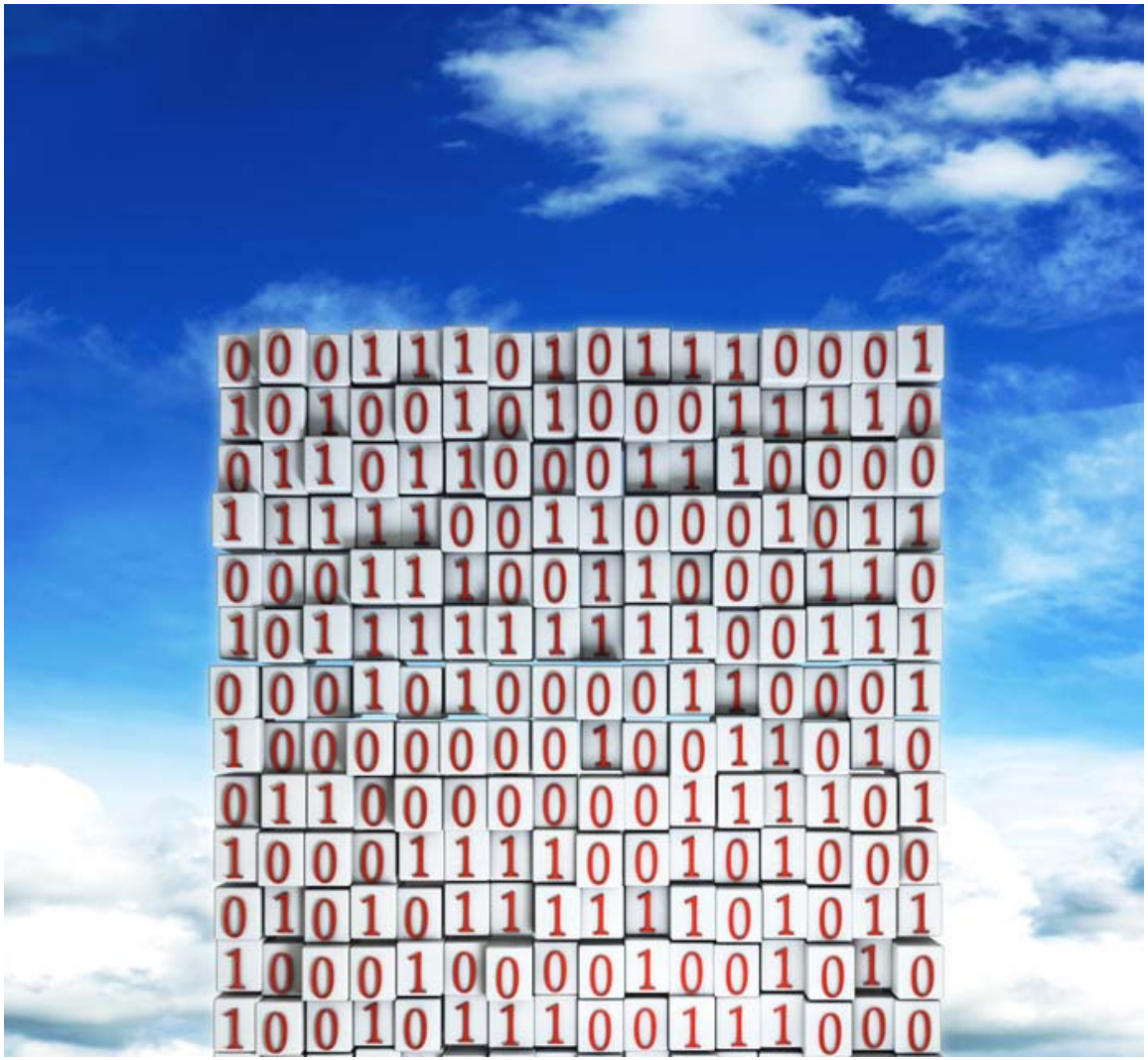
```
surface2
```

Accessing Pixels in Surfaces

Pixel

- Red
- Green
- Blue
- (Alpha)

Surface in Memory

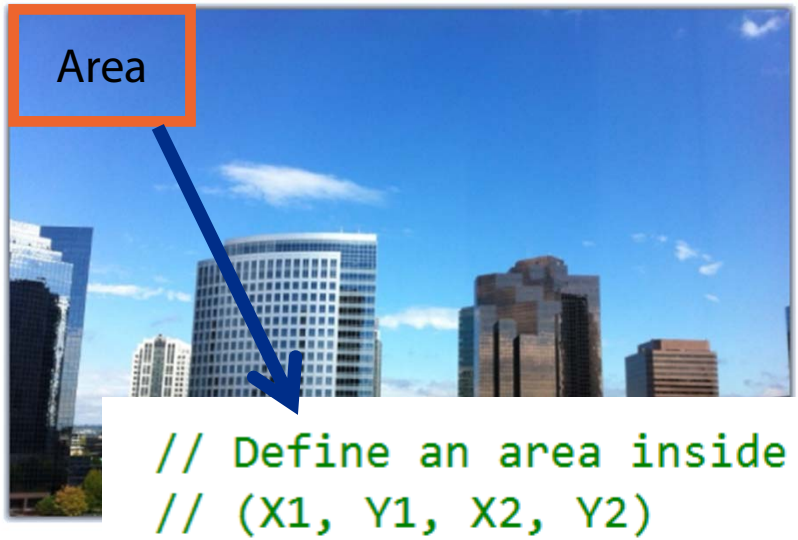


Surface Iterators

```
// Get an iterator object to the whole surface
Surface::Iter iter = surface.getIter();

while (iter.line()) {        // For each line of the surface
    while (iter.pixel()) {   // For each pixel in that line
        /*
            Manipulate current pixel's components using:
            iter.r()
            iter.g()
            iter.b()
         */
    }
}
```

Iterating the Pixels of a Surface's Area




```
// Define an area inside the surface
// (X1, Y1, X2, Y2)
Area area(0, 0, 200, 200);

// Access pixels inside that area
Surface::Iter iter = surface.getIter(area);
```

Demo: Surface R,G,B Components

Demo: Texture R,G,B Components

Demo: Slideshow

Summary

- **Loading and displaying images**
  - gl::Texture
  - loadImage()
- **Resources**
- **The Surface class**
  - Surface vs. gl::Texture
  - Surface manipulation