# Semi–optional* Programming Assignment #1

CS 200, Fall 2024

*Due Friday, December 6*

*(*) You must do at least* **one** *of the semi–optional assignments.*

## Textured mesh interface

For a textured mesh, we need to associate texture coordinates to each mesh vertex. We will do this by subclassing the `Mesh` class. The header file `TexturedMesh.h` contains the following declaration.

```
namespace cs200 {

  struct TexturedMesh : Mesh {
    virtual const glm::vec2* texcoordArray(void) const = 0;
    virtual const char *textureFileName(void) const = 0;
  };

}
```

(the file `Mesh.h` has been included). Here `texcoordArray` simply returns a pointer to an array of floating point pairs, where the $i$–th array entry gives the texture coordinates of the $i$–th mesh vertex. That is if `tm` is a `TexturedMesh` object, then

```
glm::vec4 P = tm.vertexArray()[i];
glm::vec2 uv = tm.texcoordArray()[i];
```

will retrieve the texture coordinates $(uv.x, uv.y)$ of vertex $P = (P.x, P.y, 0, 1)$. The remaining function `textureFileName` simply returns the name of the bitmap image file that is intended to be used for texturing the mesh.

As an example of this interface, I will provide you with the files `SquareTexturedMesh.h` and `SquareTexturedMesh.cpp` for a simple textured square mesh.

## Task #1: Make a tiled textured mesh

I will provide you with the bitmap file `tile.bmp`, which contains a $512 \times 512$ image that is to be used a tiled textured. Your task for this assignment is to modify your `MyMesh` class to create a textured mesh that uses the given bitmap image as its texture. The requirements are:

- The mesh should be named `MyTiledMesh`, and must be derived from the `TexturedMesh` class.

- The texture placed on the mesh should be tiled, with at least 4 copies of the given bitmap image. The tiles should preserve the aspect ratio (1:1) of the image.
- The tilings may be aligned along the object space axes of the mesh. If you decide to have the tiles rotated with respect to axes, the tiles should still be squares.

You are allowed to hard–code the mesh vertices, edge, faces, and texture coordinates. However for the texture coordinates, I would suggest that you initialize them in the class constructor, using an object–to–texture transformation.

**What to turn in**
For this portion of the assignment, you should submit two files: (1) a header file named `MyTiledMesh.h`, and (2) an implementation file named `MyTiledMesh.cpp`. You may include the header files `TexturedMesh.h`, `Affine.h`, as well as any standard `C++` header file.

# Task #2: Make a non–tiled textured mesh

I will also provide you with another bitmap image file, named `textures.bmp`. This bitmap is a single $2046 \times 764$ image that contains a collection of eight separate sub–images.

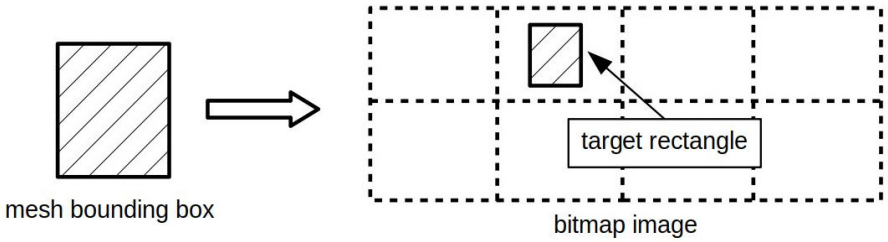
You will make another modification of your `MyMesh` class to produce a textured mesh that uses this image file as its underlying texture.

- The mesh should be named `MyTexturedMesh`, and must be derived from the interface class `TexturedMesh` class.
- The texture must be placed on the mesh in such way that mesh components must lie exclusively within one of the sub–images of the bitmap. In particular, if your mesh has only one component, the mesh should be rendered using only *one* sub–image.
- The texture placed on the mesh should preserve the aspect ratio of the given bitmap image. There should be no stretching or warping of the texture image.

Again, you are allowed to hard–code the mesh vertices, edges, faces, and texture coordinates. And again, I would suggest that you initialize the texture coordinate array in the class constructor using an object–to–texture transformation matrix.

As a hint on constructing this matrix, you should first decide how you want to map the bounding box of your mesh onto the bitmap image. Do this in bitmap coordinates. I.e., find the bitmap coordinates of the center or corners of the target rectangle in the bitmap image.

And choose the width and height of the target rectangle to match the aspect ratio of the bounding box of your mesh. You can use an image editor to help you determine these.




Once you have done this, you can convert the bitmap coordinates to texture coordinates. You are to assume that the underlying bitmap image used for texturing is the given image, and in particular, is guaranteed to be $2046 \times 764$ in size.

**What to turn in**

For this portion of the assignment, you should submit two files: (1) a header file named `MyTexturedMesh.h`, and (2) an implementation file named `MyTexturedMesh.cpp`. You may include the header files `TexturedMesh.h`, `Affine.h`, as well as any standard `C++` header file.