COSC422 Advanced Computer Graphics
Programming Exercise 7
Sprite Animation

This programming exercise demonstrates the application of point sprites in generating a sprite based animation. The DevIL library is used to load a sprite sheet and to extract frames using frame data stored in a file.

SpriteAnim.cpp:

The program `SpriteAnim.cpp` uses the sprite sheet "Runner.png" and extracts the first frame at position (24, 200) from the image and displays that frame (Fig.1). The sprite sheet contains 14 frames representing a run sequence.

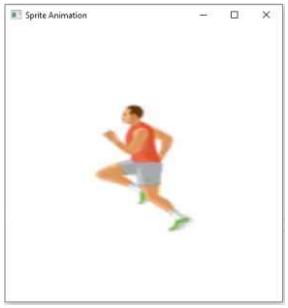

Fig. 1.

The program uses only one VBO that contains a single vertex at the origin. The `display` function just draws this point. The vertex shader (`SpriteAnim.vert`) sets the size of this point to 200.

The fragment shader (`SpriteAnim.frag`) maps the frame image to this point, using `gl_PointCoord` as texture coordinates. This mapping does not preserve the aspect ratio of the frame (the rectangular image is mapped to a square region). Please include the method given on Slide [3]-18 to correctly map the frame image to a region with the same aspect ratio. The output of the program should now change as given in Fig. 2.

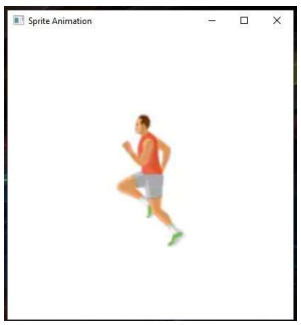

Fig. 2.

Please modify the program as detailed below to read the frame data, extract all frames and to display the sprite animation sequence.

Convert the variables `frameX`, `frameY`, `frameWid`, `frameHgt` and `aspectRatio` into arrays of size `NUMFRAMES`. Include the necessary statements in the `initialize()` function to read data from the input file "Runner_Frames.txt" and store them in the above arrays.

In the function loadTextures(), use a for-loop to generate textures corresponding to each frame. The variable `texID` should also be converted into an array. Associate each texture with the corresponding active texture units (Note: GL_TEXTURE3 can be written as GL_TEXTURE0+3).

The uniform variables "aspect" and "spriteTex" in the fragment shader must also be converted into arrays of size 14. The values of a uniform float array are passed to the fragment shader as shown below:

```
glUniform1fv(aspectLoc, NUMFRAMES, aspectRatio);
```

In a similar manner, the uniform integer array "spriteTex" must be assigned numbers 0, 1, 2, ..NUMFRAMES-1 (these are active texture unit indices).

Please include a timer callback function that updates the frame index from 0 to NUMFRAMES-1 in a continuous repeating cycle. This frame index must also be passed to the fragment shader as a uniform integer value. The fragment shader uses the frame index to retrieve the corresponding aspect ratio and texture object from the arrays, for displaying the current frame.

The above steps if implemented correctly, should produce the display of the animated sprite.