Computer Science

COMPSCI 373 S1 C 2015

Making a Ray Caster: Part 2

The work done on this task must be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone else for help. Under no circumstances should you work together with another student to solve problems posed in this task.

Assignment

This week's assignment will be composed of two Quizzes in coderunner. The theoretical Quiz will have 9 questions, and one feedback question, each worth 0.25 marks. You will have 1 hour (and 3 repeats) to answer it. The programming Quiz will have 4 small programming/practical questions each worth 0.25 marks. You will have 2 hours (and 3 repeats) to answer it. Both theoretical and practical questions will query about the implementation of a Ray Casting engine based on the 2D Ray Casting section of your lectures. This document proposes a step-by-step implementation of a ray caster in line with the supporting material provided. It should help you prepare for the programming Quiz and better understand the principles of a Ray Caster. It is up to you to complete the below tasks. There is no grade attached to the completion of the skeleton code provided here.

Included resources

Please see Part 1 of the supporting material. This document assumes you have completed stages 1 to 6, which are described in the earlier supporting material.

Stage 7: Calculating wall strip heights

Your first step in developing the rendering component is to write the function which, when given a ray length, calculates the height of the corresponding wall strip which is drawn to the screen (you do not have to deal with distortion here).

The function can be found in **renderer.c**, and has the following signature:

```
float calculateWallColumnHeight(float rayLength)
```

When completed, the function should return the correct wall strip length.

Stage 8: Drawing untextured walls

Your next step is to complete the function which renders an untextured wall column to the screen. The function you need to complete is in **renderer.c** and has the following signature:

```
void drawUntexturedWallColumn(int x, float wallYStart, float length, Uint32 ABGRColor,
float illumination)
```

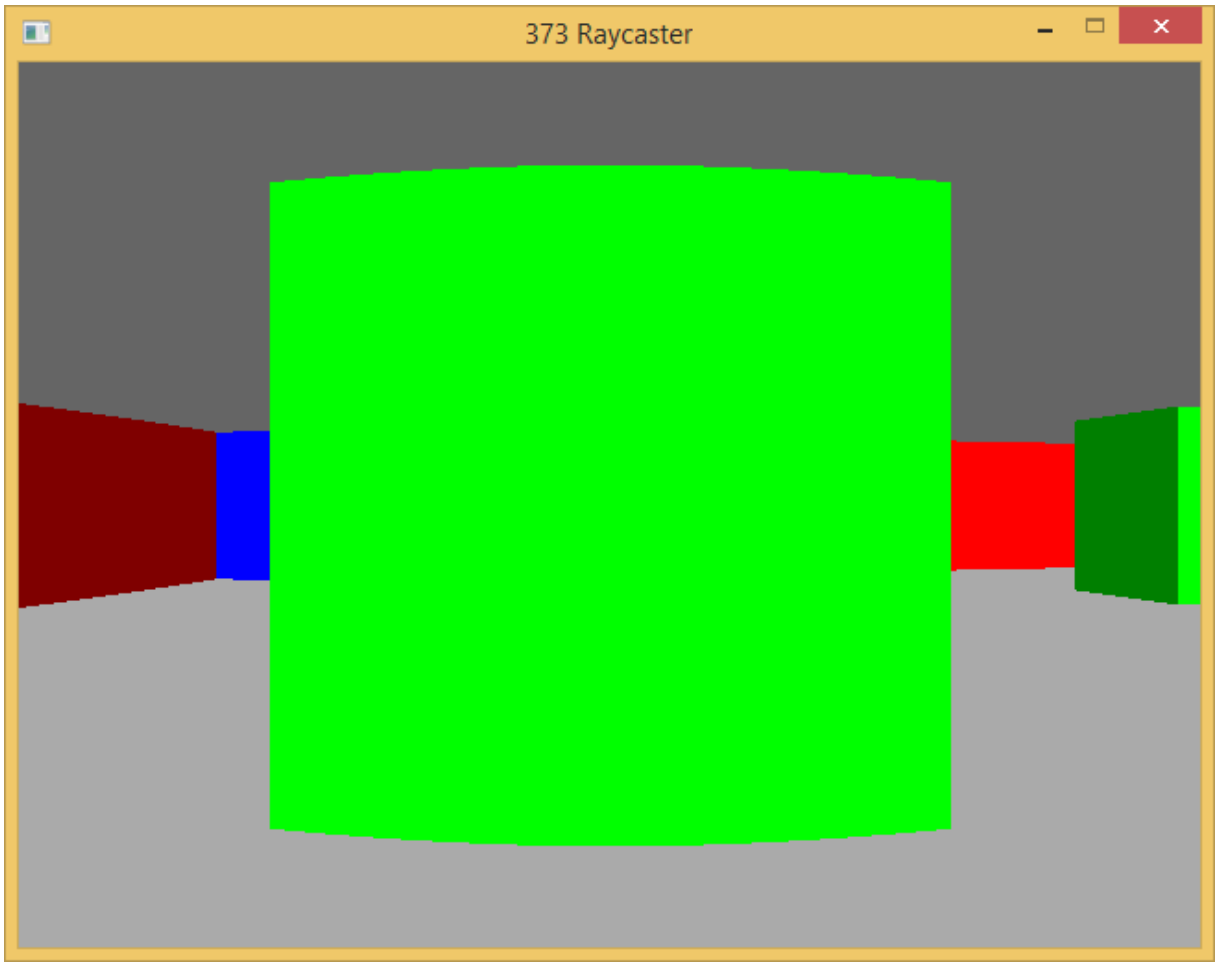
A description of the input parameters:

`int x` - The x screen (viewport) coordinate of the pixels to draw
`float wallYStart` - The y coordinate to start drawing the strip at (everything above is the ceiling)
`float length` - The length of the wall strip to draw
`Uint32 ABGRColor` - The Color integer to use (after it has been shaded)
`float illumination` - The darken/lighten factor

Most of the function has already been implemented, your job is to implement the lines which set the appropriate pixels for the floor and wall parts of the pixel strip. The code that sets the ceiling pixels are provided.

The floor color is given in a global variable named FLOOR_COLOR.

Once you have completed this function, run your program and press 'm'. You should see a view like the following:



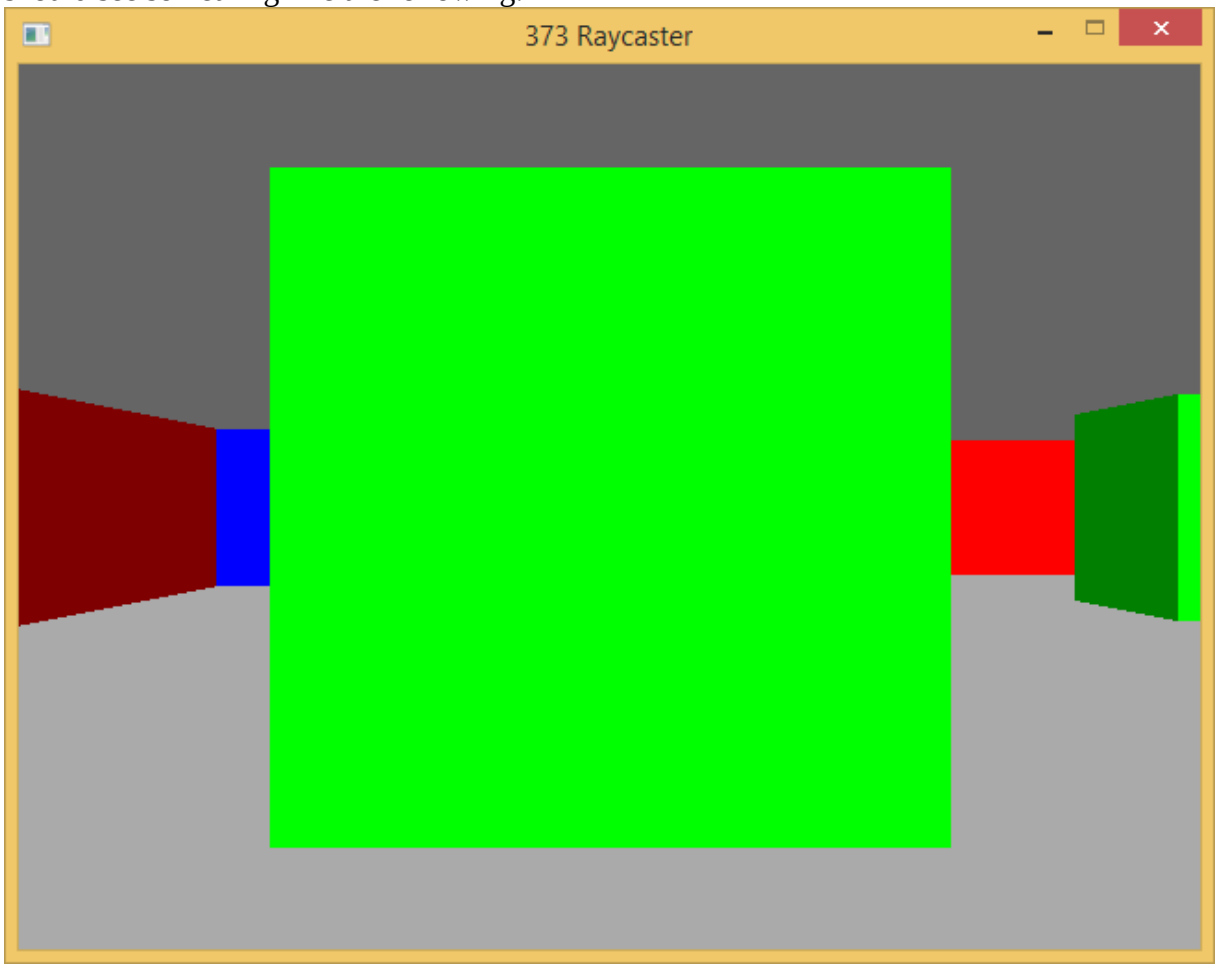
Once you are satisfied that your solution is correct, you can move on to *stage 9*.

Stage 9: Removing distortion

Your goal in this stage is to provide the function to produce the undistorted length of a ray (as per your lecture slides and tutorial notes on undistortion).
The function is found in **renderer.c**, and has the signature:

```
float getUndistortedRayLength(Ray ray)
```

Using the given ray, you must derive and return the undistorted length of the ray. Once you have completed this stage, you can run your program, and then press ‘m’ followed by ‘f’. You should see something like the following:



Once you are satisfied that your implementation is correct, you can move on to *Stage 10*.

Stage 10: Calculating the texture column

Now you are ready to start implementing texture mapping. The first function you must implement for this one which will calculate the texture column for a given intersection. This function is provided in **renderer.c**, and has the signature:

```
int getTextureColumnNumberForRay(Ray ray, Vector3f rayStart)
```

You are given the ray, as well as the start position of the ray (the camera coordinate). The first step in implementing this function is to derive the hit position in the world from both the camera position, and the given ray. Once you have this, you can use the procedure discussed in tutorials and lectures in order to derive the texture column number. The code to reverse the texture in the appropriate cases is provided to you.

Once you are happy with your implementation, you can move on to *Stage 11*.

Stage 11: Drawing textured walls

The final task for completing the full Ray Caster is to implement the function which draws a textured wall strip. The function is similar to the function to draw an untextured strip, but with some different parameters:

```
Uint32* texture     - The texture to use for the wall
int textureCol      -  The texture column number to use for this strip
```

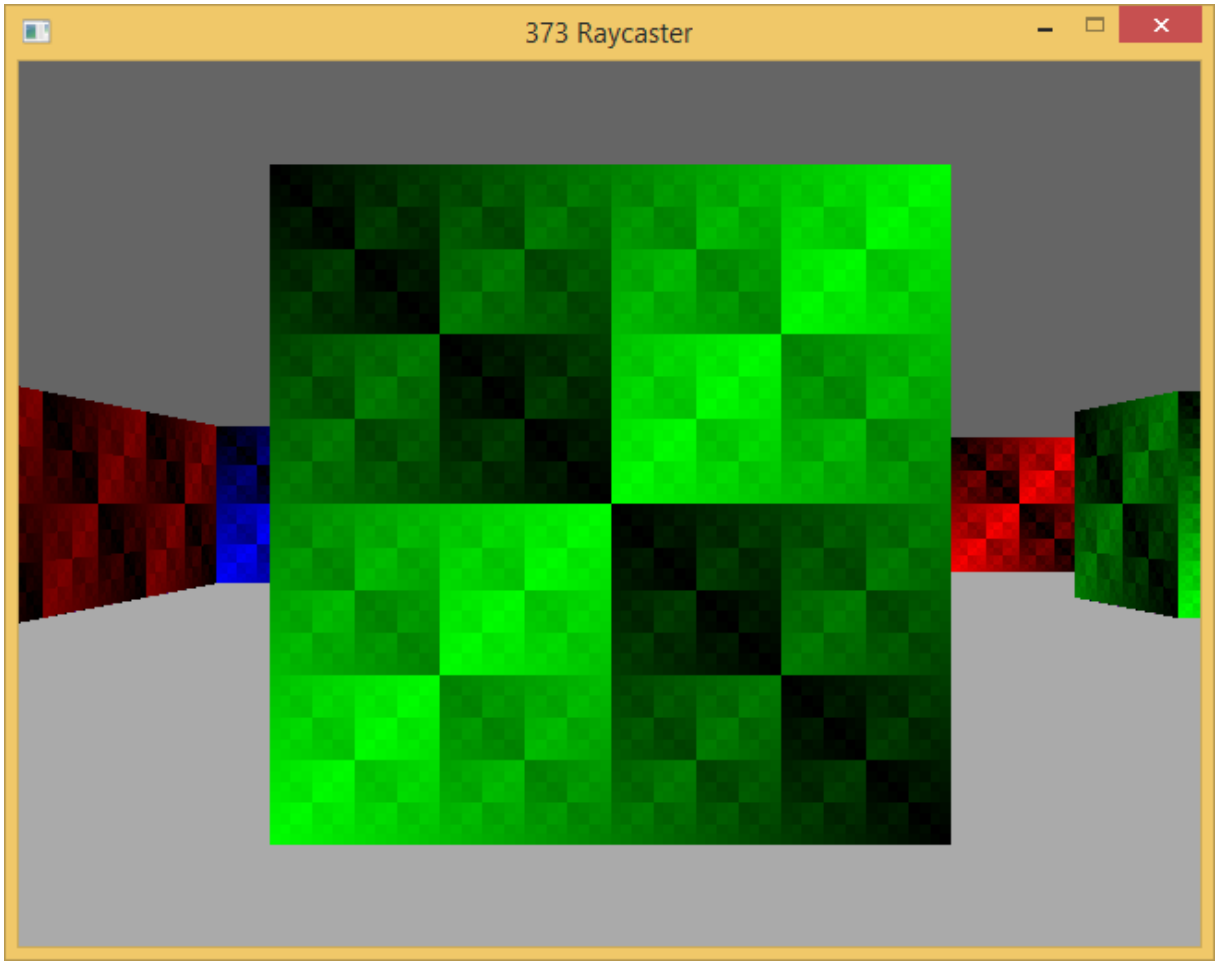
The main task of this function is to scale a given texture strip to the screen. The linear scaling code is provided for you, and you can use the variable *ty* as the y coordinate of the texture pixel currently being drawn.

You will need to provide the code for drawing the floor and ceiling pixels (it is the same procedure as for untextured walls).

You must also provide the code to extract a texture color value for the current texture pixel being drawn. The code for this is provided, apart from the x and y coordinates of the texture pixel to retrieve.

Finally, the retrieved color should be passed to the *shadeColor* routine, and then drawn to the screen at the appropriate location.

Once you have completed this function, if you run your program and press ‘m’ followed by ‘f’ and ‘t’, you should see the following:



If you see something like this, congratulations! You have completed the textured Ray Caster.

Appendix

Please see Part 1 of the assignment supporting material.