Department of Computer Science and Engineering

Course Code: CSE 423	**Credits: 1.5**
Course Name: Computer Graphics	**Semester: Spring 24**

Lab 03

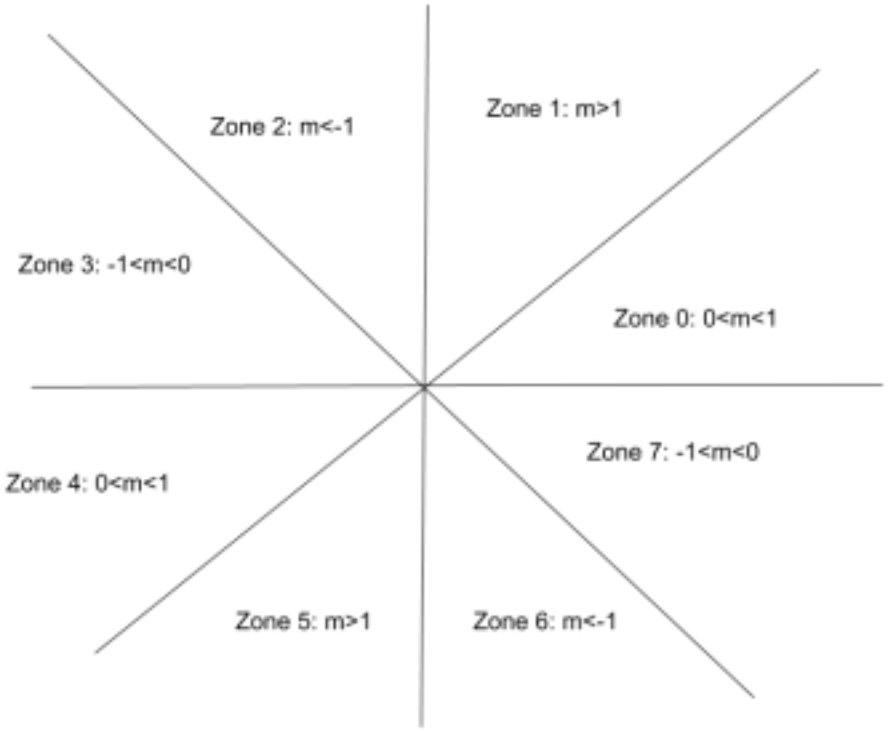
Midpoint Circle Drawing Algorithm

I. Topic Overview:

The students were introduced to the Midpoint line drawing algorithm in the previous class. Given the coordinates of any two points on the line: (x1, y1) & (x2, y2), the student's task was to find all the intermediate points required for drawing the line on the computer screen of pixels where every pixel has integer coordinates.

Today, we would be introducing a similar approach to draw a circle. This algorithm tries to find the best approximation of the next point on the circle using a decision parameter. As we will only choose one between two pixels as we did previously in the midpoint line drawing algorithm it will be a quite fast approach.

We would be drawing a part of the circle, let's say only zone 0. The students might think that they might have to generate 8 different equations for eight different zones. However, we will be applying the 8-way symmetry approach to draw the portion of the circle for the rest of the zones.



II. Lesson Fit:

a. Implementation of Midpoint circle drawing algorithm using 8-way symmetry. b. The students should have a clear understanding of the 8-way symmetry approach. c. They should also have a clear idea about all the different zones.

III. Learning Outcome:

After this lecture, the students will be able to:

a. Learn how to use a decision parameter to determine the next pixel on the circle. b. Learning how an 8-way symmetry approach naturally helps to draw a circle.

IV. Anticipated Challenges and Possible Solutions

Students might get confused about why and how we are using the 8-way symmetry approach.

Solutions:

Students should understand why and how we are drawing a point of a particular point to another zone and in which zone the calculations are being done for drawing the circle.

V. Acceptance and Evaluation

Students will start implementing the algorithm after we finish our lecture. If a student fails to complete the implementation he/she will have to complete it by that night and show it to the class teacher during his/her consultation period. If a student also fails to do that then instead of 10 we will evaluate the student on 7.

1. Lab evaluation marks: **out of 10**
2. Late Lab evaluation marks: **out of 7**

VI. Activity Detail

a. **Hour 1:**

Discussion:

A circle is defined as a set of points that are all at a given distance r from a center positioned at (x_c,y_c).

So, the equation of circle according to the center (x_c,y_c) is,

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad \text{..... 1}$$

from equation 1 we get the value of y as below,

$$y = y_c \pm \sqrt{r_2 - (- x_{c2}\ x)} \quad \text{..... 2}$$

As $f(x, y) = 0$ represents a circle, Let,

$$f(x, y) = (x - x_c)\ y) \text{ ..... } 3^{\ 2} + (- y_c^{\ 2} - r^2$$

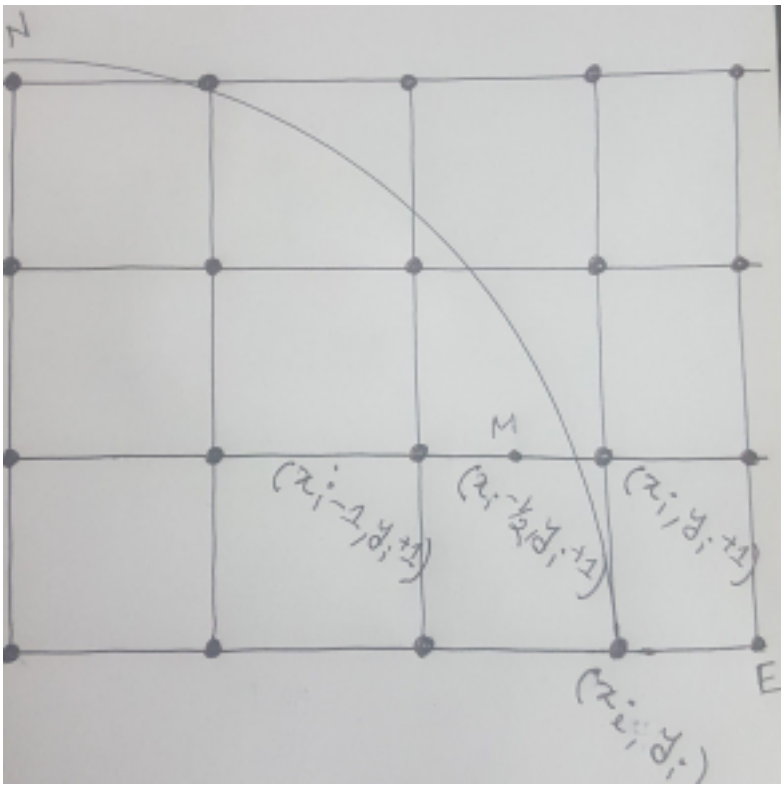
Now, for any point,

$f(x, y) = 0$, the point is on the circle

$f(x, y) > 0$ **,** the point is outside the circle

$f(x, y) < 0$ **,** the point is inside the circle

In the midpoint circle drawing algorithm at each of the i^{th} steps we will be evaluating the circle function to come to a decision. Let us assume that we are giving unit increments to x in the plotting process and determining the y position using this algorithm. Assuming we have just plotted the i^{th} pixel at (x_i,y_i), we next need to determine whether the pixel at the position (x_i+1,y_i) or the one at (x_i,y_i+1) is closer to the circle.



Our decision parameter d_i at the i^{th} step is the circle function evaluated at the

midpoint of these two pixels. The coordinates of the midpoint of these two pixels

$_i - {}_2^1 y_i + 1$

is $(x\ ,\ )$.

Hence, $d\ (x\ ,\ )\ x\ )\ y\ )$ …….. $4\ _i = f_i - {}_2^1 y_i + 1 = (_i - {}_2^1{}^2 + (_i + 1^2 - r^2$

Successive decision parameters are obtained using incremental calculations, thus avoiding a lot of computation at each step. We obtain a recursive expression for the next decision parameter.

$$_{i+1} = f_{i+1} - {}_2^1 y_{i+1} + 1 = (_{i+1} - {}_2^1{}^2 + (_i + 1 + 1^2 - r^2$$

Therefore, $d\ (x\ ,\ )\ x\ )\ y\ )\ \ d\ x\ )\ y\ )$ …….. $5\ _{i+1} = (_{i+1}$ $- {}_2^1{}^2 + (_i + 2^2 - r^2$ Now we get by subtracting eqn (4) from eqn (5)

$$\Delta d = d_{i+1} - d_i$$

$$_{i+1} - {}_2^1{}^2 - (_i - {}_2^1{}^2 + (_i + 2^2 - (_i + 1^2 - r^2 + r^2$$

$$\Delta d = (x\ )\ x\ )\ y\ )\ y\ )$$

$\Delta d = (x\ ).(x\ )\ y$ …… $6\ _{i+1} + x_i - 1\ _{i+1} - x_i + 2\ _i + 3$ Now if $d =$, then the midpoint of the two possible pixels lies within the circle, $_i < 0$ thus the north pixel is nearer to the theoretical circle.

Hence, x . Substituting this value of in eqn (6), we get, $_{i+1} =$

$x_i\ \Delta d = d\ x\ ).(x\ )\ y\ _{i+1} - d_i = (_i + x_i - 1\ _i - x_i + 2\ _i + 3$

$d\ x\ ).0\ y\ _{i+1} - d_i = (_i + x_i - 1 + 2\ _i + 3$

$d\ y\ _{i+1} = d_i + 2\ _i + 3$

And if we consider d then, the midpoint of the two possible pixels lies $_i > 0$ outside the circle, so the north west pixel is nearer to the theoretical circle.

Therefore, x . Substituting this value of in eqn (6), we get, $_{i+1} = x_i - 1\ \Delta d = d\ x$ $).(x\ )\ y\ _{i+1} - d_i = (_i - 1 + x_i - 1\ _i - 1 - x_i + 2\ _i + 3$

$d\ 2x\ ).(-\ )\ y\ _{i+1} - d_i = (_i - 2\ 1 + 2\ _i + 3$

$d\ x\ y\ _{i+1} = d_i - 2\ _i + 2 + 2\ _i + 3$

$d\ x\ y\ _{i+1} = d_i - 2\ _i + 2\ _i + 5$

For the initial decision parameter $x = r$, $y = 0$. So, from eqn (4) we get, $d\ r\ )\ 0\ )\ _0 = (\ - {}_2^1{}^2 + (\ + 1^2 - r^2$

$d_0 = \frac{5}{4} - r$

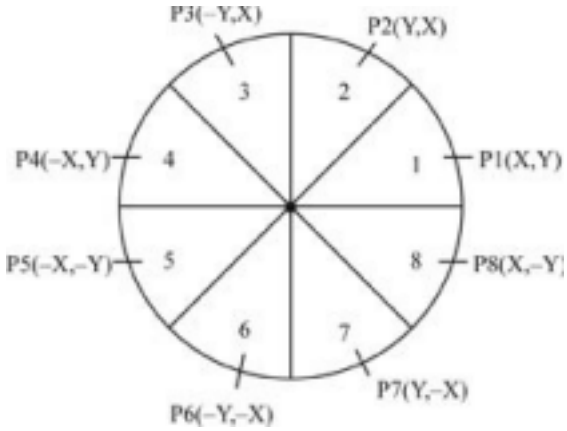
To get the integer value of pixel coordinates, we approximate,

d …… $7_0 = 1 - r$

b. **Hour: 2**

Discussion: 8-way Symmetry

Today we will go through the 8-way symmetry approach and we will discuss how we can apply this approach to draw a circle easily.



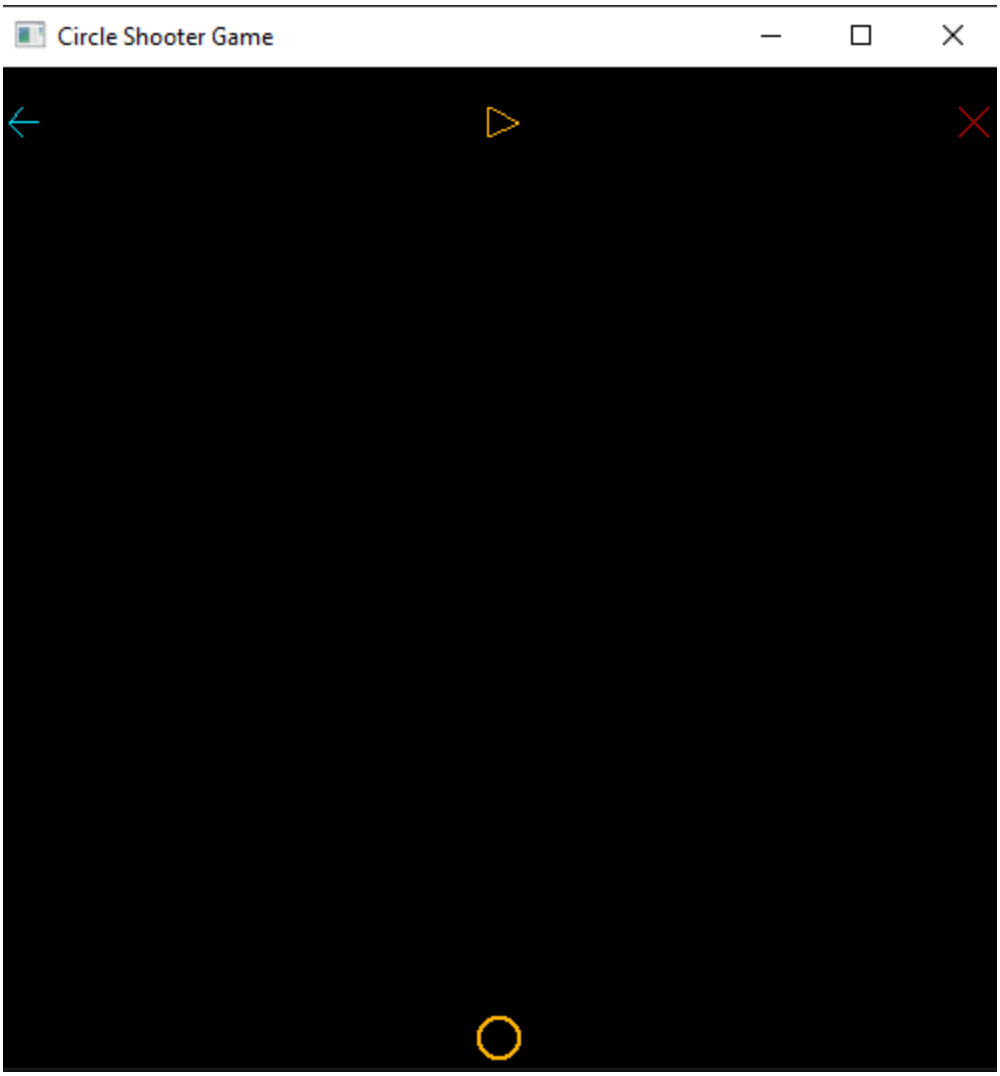
c. **Hour: 3**

Discussion: Implementation

Now we will be implementing the midpoint circle algorithm in the lab.

VII. Home task:

In this lab, students will create a simple 2D game called "Shoot The Circles!" from scratch. The objective is to shoot falling circles before they reach the ground. Players score points for every successful hit but must avoid missing three circles, which ends the game. The lab includes a shooter circle at the bottom, controlled with 'a' and 'd' keys for horizontal movement, and the ability to shoot upwards with the spacebar.



Game Rules:

1. **Shooter Control:** A shooter circle is placed at the bottom of the player's screen, and it can be moved horizontally using the 'a' and 'd' keys. The shooter's movement is restricted to stay within the screen boundaries.
2. **Firing Mechanism:** Players can shoot by pressing the **spacebar**, sending a fire projectile upwards on the screen.
3. **Falling Circles:** Circles fall vertically from the top of the screen. To score a point, the shooter must hit a falling circle directly. The horizontal position of each falling circle **is** randomized.
4. **Scoring:** Successfully hitting a falling circle increases the player's score by 1 and both the projectile and falling circle will be removed from the screen upon hit. The current score is displayed in the console for tracking.
5. **Game Over:** The game can be over in any of the first two following ways:
    i. If a falling circle touches the circle shooter directly, the game will be over immediately.
    ii. If the player misses three falling circles to shoot, i.e. falling circles crossing the bottom boundary before the circle shooter manages to vanish them three times, the game will be over.
    iii. Bonus and Optional: The game will also be over if a player misfires, i.e. shoots but fails to hit any falling circle three times, the game will be over.

    In this state, falling circles disappear, shooter movement is disabled, and "Game Over" is displayed in the console along with the final score.
6. **Control Buttons:** Three clickable buttons are positioned at the top of the screen:
    a. Left Arrow (Restart): Clicking this button restarts the game, resetting the score and circle speed. A message like "Starting Over" is shown in the console.
    b. Play/Pause Icon (Amber Colored): This button toggles the game between play and pause states. The icon represents the current state. In the pause state, falling circles freeze, and shooter movement is disabled.
    c. Cross (Red Colored): Clicking this button prints "Goodbye" in the console along with the final score and terminates the game.

7. **Circle Drawing:** All circles on the screen are drawn using the midpoint circle drawing algorithm. The primitive type used for drawing circles is GL_POINTS. The size of the circles is designed to be visible but not overly large, similar to the shooter's size.

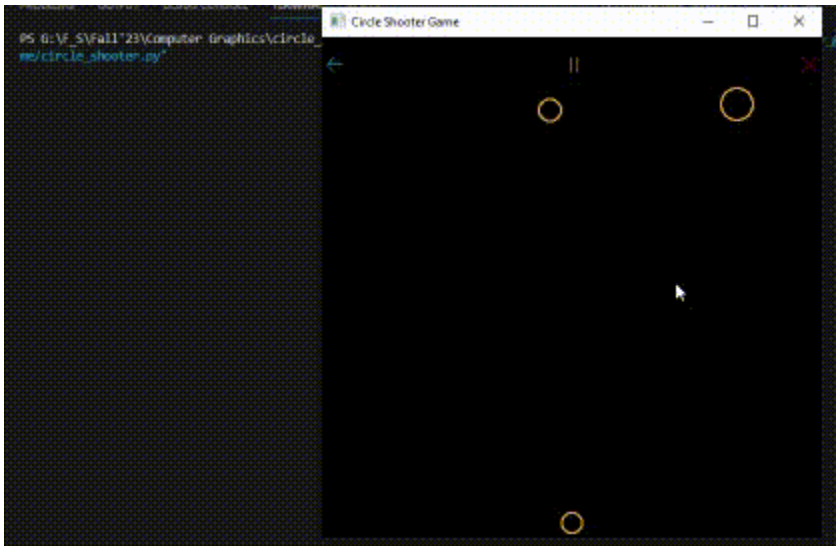


Fig: Shoot the Circles, Basic Gameplay

Please Note: The above figure is an animated gif. If it doesn't play, you might want to open it in Google Docs instead of MS Word.