GAME REQUIREMENTS

Background/Graphics:

Add Moving Patterns to Game

We make "Ping Pong” interesting by adding patterns that move in the background. These

patterns change like in the games Temple Run or Maze Runner. They will keep moving in the

background to give an illusion as if the background is moving while playing game, They make

the game look nice but do not change how the game is played.

• Background Layers: Use different layers of simple patterns that scroll. Each layer

scrolls at its own speed. This makes the background seem alive while playing.

• Types of Patterns: Use patterns made from characters to show different styles like

stripes, dots, or waves.

• Using Patterns Right: Make sure the game does not slow down because of new

patterns. Use patterns that are easy to draw and erase in memory without stopping the

game.

• Choices for Players: Let players choose if they want moving patterns or not before

starting the game.

• Managing Patterns: Learn how to draw and move patterns in video memory using

NASM. This includes using loops to draw lines and shapes.

• Keep Game Fast: Use small and simple patterns.

• Control with Keys: Allow players to press a key to turn patterns on or off. You might

use a simple key like 'P' for patterns. Check if this key is pressed and show or hide the

patterns.

1. Player Controlled Paddles

The game will have two paddles, one for each player.

• Right Paddle (Player 1): Controlled by the 'W' and 'S' keys (for up and down

movement, respectively).

• Left Paddle (Player 2): Controlled by the Up Arrow and Down Arrow keys (for up and

down movement, respectively).

Instructions:

Implement key press detection: Use the INT 16h to capture key presses for controlling

the paddles.

o 'W' = 0x11

o 'S' = 0x1F

o Up Arrow = 0x48

o Down Arrow = 0x50

Paddle Movement: Each paddle is controlled by updating its vertical position on the

screen

o To move up, decrement the position.

o To move down, increment the position.

o Ensure paddles stay within screen bounds (e.g., the top-most row is position 0,

and the bottom-most row is position 24 for a 25-row screen).

Paddle Movement: Each paddle must 3 characters tall, with the paddle position

representing the middle part of the paddle.

← Paddle

← Actual Paddle Position

← Paddle

Display the paddles: Each paddle can be represented by a character such as a |

2. Ball Movement

The ball moves around the screen and reflects off walls and paddles.

Instructions:

1. Ball Starting Position: Place the ball at the center of the screen (e.g., at position (12,

40) for a 25x80 screen).

2. Direction: The ball can move in any of the four diagonal directions. This can be

represented by one direction variable as a vector.

3. Ball Movement: Each frame, update the ball's position by adding the direction variable

to the balls current position.

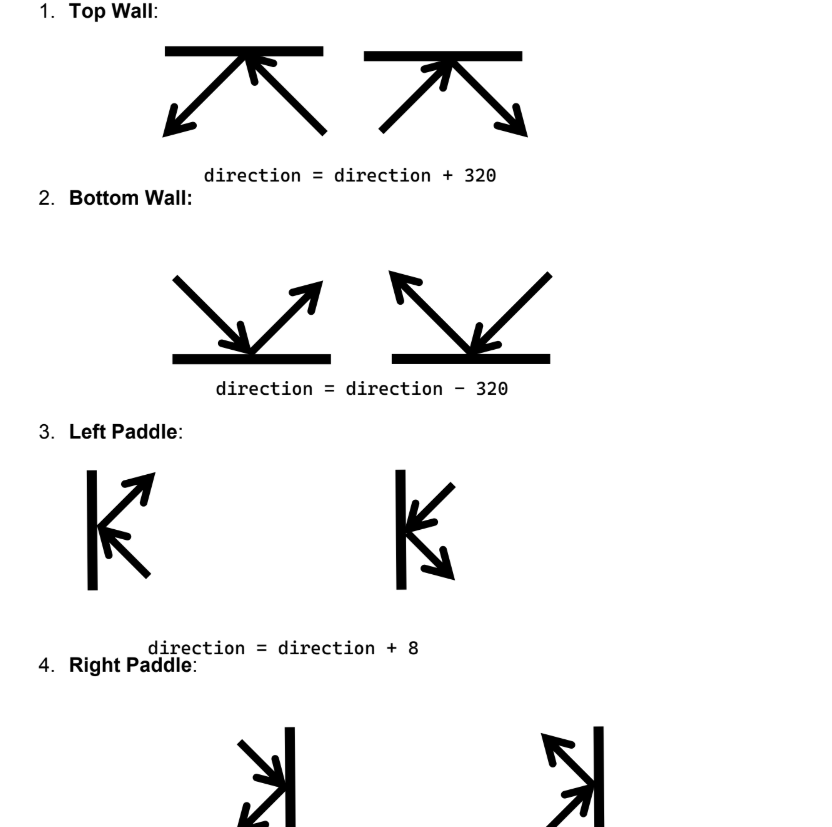
4. Draw the Ball: After updating the ball's position, display the ball on the screen.

Represent the ball as a simple character, such as O.

3. Reflection Logic

When the ball hits a wall or a paddle, it should reflect (bounce back). To accomplish this,

update the direction variable after each collision.



4. Score Counting

The game should keep track of the score for both players.

Instructions:

Increment Score: Each time the ball passes a player’s paddle (i.e., crosses the screen

boundary), the other player scores. To detect whether or not the ball has crossed the

boundary get the ball position:

o If (position % 160 == 0) then ball has crossed the left boundary. In this case

Right player’s score increments by 1.

o If ( (position+2) % 160 == 0 ) then ball has crossed the right boundary. In this

case Left player’s score increments by 1.

Display the Score: Show the current score at the top of the screen.

Winning Condition: The game ends when one player reaches a pre-set number of

points (e.g., 5 points). Display a message indicating the winner.

5. Game End

The game should have a winning condition and display the winner.

Instructions:

Winning Score: Before the game starts, allow the user to choose the score needed to

win (e.g., 5 points).

Game Loop: Continuously update the game state until one player reaches the winning

score.

Display Winner: When a player wins, display a message such as “Player 1 Wins!”.

Then, allow the user to restart or exit the game.

6. Pause/Unpause

The game should allow the user to pause and unpause at any time.

Instructions:

Pause Key: Choose a key to pause and unpause the game. For example, use the P

key.

Pause Logic: When the game is paused, stop the movement of both paddles and the

ball. The ball and paddles should be frozen on the screen.

Unpause Logic: When the user presses the pause key again, resume the game from

where it was paused.

7. Board Printing

At each iteration of the game loop, clear the screen and redraw the entire board along with the

graphics in the background (explained in start)

Instructions:

Screen: First clear the whole screen by placing a black character or empty space on

each video memory location.

Walls: Then print the top and bottom walls.

Paddles: Then print the paddles at the stored paddle position variables.

Ball: Then print the ball at the stored ball position variable.