

Ethan Alward ID: 3695438

February 4th, 2025

Concept Assignment #1

CS2053 Introduction to Game Development

1. Provide pseudo-code similar to the pseudo code found in the book (and slides) for Pac Man but do it for the first level of the classic game Pong! See a video of Pong! here to remind yourself

Not sure how much detail you’d like here, I assume you want to see the gameloop code.

Runs once before loop starts looping:

#create UI, Paddles, Ball

#load the paddles and ball from their scenes

var player1 = load(“res://paddle.tscn”) #create the first player

var player2 = load(“res://paddle.tscn”) #create the second player

var ball = load(“res://ball.tscn”) #create the ball

#set up the moveable nodes

player1.instantiate()

player2.instantiate()

#these above calls would set user keyboard inputs(up arrow set player up) and set paddle’s initial position

ball.instantiate()

#this would set the ball’s initial position, velocity, and direction

#set up UI

$UI.DashedLine.visible = true

player1Score = 0

player2Score = 0

$UI.Player1Score.text = “0”

$UI.Player2Score.text = “0”

$UI.Player1Score.visible = true

$UI.Player2Score.visible = true

#set winning score

maxScore = 10

#set game space to be 640x480

player1Scored = 641

player2Scored = -1

Game Loop Starts:

#check if player scored, if score is at maxScore end game otherwise keep ball moving

if ball.position.x <= player2Scored:

ball.visible = false

ball.position.x = 320

player2Score++

if player2Score == maxScore:

$UI.endgame.visible = true

wait(5)

for child in main.get\_children()

child.queue\_free()

else:

if ball.position.x >= player1Scored:

ball.visible = false

ball.position.x = 320

player1Score++

if player1Score == maxScore:

$UI.endgame.visible = true

wait(5)

for child in main.get\_children()

child.queue\_free()

#keep ball moving correctly

#in the video it appears that on contact the ball’s direction is the same as the paddle (hitting up, ball goes up) the speed of the paddle also affects the speed of the ball (slide up quick, ball moves quick)

#handle paddle movement

player1.update\_position\_and\_velo()

player2.update\_position\_and\_velo()

#in the following code I would actually use bounce(move\_and\_collide(velocity\*delta).get\_normal())

#in contact with paddle

If ball.position.x < 20 and ball.position.x > 10:

if player1.position.y == ball.position.y:

#account for contact

#ball speed correlates to player paddle speed

ball.velocity.x = -(ball.velocity.x \* player1.velocity.x \* delta)

ball.direction.x = -ball.direction.x

#direction of ball goes the way the paddle does (hit up, ball goes up)

ball.velocity.y = player1.velocity.y \* delta + ball.velocity.y

ball.direction.x = -ball.direction.x

else:

If ball.position.x > 620 and ball.position.x < 630:

if player1.position.y == ball.position.y:

#account for contact

#ball speed correlates to player paddle speed

ball.velocity.x = -(ball.velocity.x \* player1.velocity.x \* delta)

#direction of ball goes the way the paddle does (hit up, ball up)

ball.velocity.y = player1.velocity.y \* delta + ball.velocity.y

ball.direction.x = -ball.direction.x

else: #let ball movement continue

#if ball hits bottom send it up

if ball.position.y == 0:

ball.direction.y = -ball.direction.y

else:

#if ball hits top send it down

if ball.position.y == 480:

ball.direction.y = -ball.direction.y

else:

#let ball propagate

ball.position.x == ball.position..x \* delta

ball.position.y == ball.position.y \* delta

2. In Godot, describe the difference between the three different event functions \_process, \_physics\_process, and \_ready.

a) What is the role of these three functions, when can they occur, and in what order?

\_ready is called when both the node and all of its child nodes have entered the scene tree / when you run the project. \_process is called every frame in the game loop while \_phyiscs\_process is called at a specified rate, initialized to 60 times a second. Ready occurs first, \_process and \_physics\_process can occur at the same time if their FPS is the same. \_physics\_process should be used when users have the ability to use different fps.

b) How do they relate to the game loop?

\_ready is called once when a node enters the scene tree, the root node’s ready function will be called first during the game loop’s first iteration. The \_process and \_physics\_process function run every frame/loop.

c) Can we control how often each of these functions occur, and if so, how?

ready can only be called once. We can specify in project settings how often we want \_process and \_physics\_process to run, we can also turn them off and on with set\_process and set\_physics\_process.

3. In 2D game graphics, explain why buffering is commonly used. Next, explain a reason why someone might want to disable buffering.

Buffering is commonly used in 2D game graphics to get the next visual frame loaded before displaying it. Buffering is just getting the next thing ready to be displayed. The main reason to do this is to prevent choppy frames and allow for smooth animations. Since buffering does cause a delay, in a situation where you need instant results it could make sense to disable buffering.

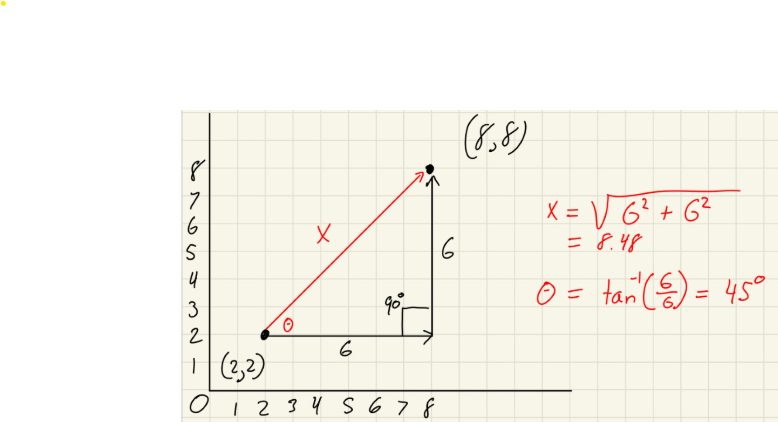
4. What is the difference between a tilemap and a tileset?

A tilemap is a node in 2D tile based maps. Tilemaps use tilesets which are groups of tiles. A tileset is essentially a collection of all of the images you would like to have in your game. Typically all of these images lie on one tile sheet, and the ones you’d like to use on a specific tilemap are cropped out and added to a tileset. Tiles are images that are added into TileSets which are added to TileMaps, TileMaps are actually what is displayed in the game.

5. Assume two game objects gameObject1 and gameObject2 in a 2D game.

a) Assume that gameObject1.position is at (2, 2) and gameObject2.position is at (8, 8), what is the vector that points from gameObject1 to gameObject2 ?

This question is confusing, from what I’ve read an object with position 2,2 already is pointing in the same direction as an object at 8,8 or an object at 100,100. If I have an object at 2,2 to get to 8,8 I’ll need a vector of 6,6. You could use scalar multiplication on gameObject1’s current vector, 2,2 \* 4 will give you 8,8 as well. To get the vector from gameObject2 to gameObject1 I could also just subtract gameObject2’s vector from gameObject1’s.



b) Write code to calculate direction from gameObject1 ’s position to gameObject2 ’s position using vector calculations. Note that direction must be a normalized vector

gameObject1.direction = gameObject1.position.direction\_to(gameObject2.position)

The direction\_to function returns a normalized vector. It is like saying (gameObject2.position - gameObject1.position).normalized

6. You are coding a top-down 2D game with stereo sound. In a cut-scene you are scripting and an explosion occurs. You must determine whether to play the sound in the right or left speaker depending on where the explosion has occurred relative to the character. Describe how you can solve this problem using vector math. Provide formulas to help explain your answer.

You want the sound to be played in the left and right speaker proportionally to how an actual explosion’s sound waves would hit your ear. On the explosion I would want to get the vector from the character to where the explosion occurred. If the explosion happened straight ahead we’d want a 50/50 split between the two speakers, if it happened directly to the left I would want 100% of the sound to go to the left speaker. Assuming position is set to be a vector relative to 0,0.

#the closer the explosion the louder it will be

volume = 1 / (explosion.position - character.position)

#now that I have a ratio for how loud the noise itself will be, I've got to find a way to split it up between speakers. The smaller the angle to the x axis the more sound the corresponding speaker will get.

if explosion.position.x < 0: #explosion occured on left side

leftSpeakerAngle = cos^-1(character.dot(explosion) / character \* explosion)

rightSpeakerAngle = 180 - leftSpeakerAngle

else:

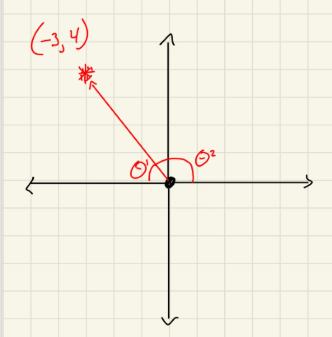
rightSpeakerAngle = cos^-1(character.dot(explosion) / character \* explosion)

leftSpeakerAngle= 180 - rightSpeakerAngle

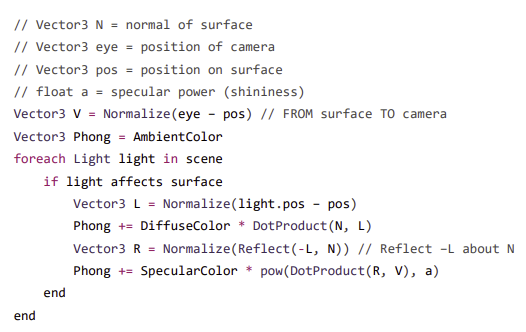
rightSpeakerVolume = (1 / rightSpeakerAngle) \* volume

leftSpeakerVolume = (1 / leftSpeakerAngle) \* volume

Here’s a little diagram I drew to make it make sense in my head, here since angle1 is smaller obviously more sound will go to it.



7. The following is the algorithm of Phong Reflection Model:



7. Based on this code:

a) In Phong += DiffuseColor \* DotProduct(N, L), what is the meaning of DotProduct(N, L)? Why do we need to multiply the value of DotProduct(N, L) with DiffuseColor? For a directional light without a light source position, what is the L vector?

A normal is a vector with a magnitude of 1, normalizing a vector distills it down to a direction. As for L, subtracting the position of the surface from the position of the light gives a vector pointing from the surface to the light. The Dot Product gives us a scalar value meaning only a magnitude. The Dot Product of the vector pointing from the surface to the light, and the normal of the surface will give a value that represents how far away the surface is from the light. Multiplying that value by the DiffuseColor now makes sense because the Phong needs to know how far away the surface is from the light to know what shade to display on the surface. L is the vector pointing from the surface to the position of the light, if the light source no longer had a position L would still need to represent the direction that the directional light is coming from.

b) For Phong += SpecularColor \* pow(DotProduct(R, V), a), what is the meaning of DotProduct(R, V)? Why we need to multiply value of pow(DotProduct(R, V), a) with SpecularColor?

This portion of the formula is used to measure how shiny the surface of an object should appear. The DotProduct(R,V) returns a value that represents how well the light reflects into the camera. We need to multiply the pow(DotProduct(R,V), a) with a SpecularColor because the specular color is different for different meshes.

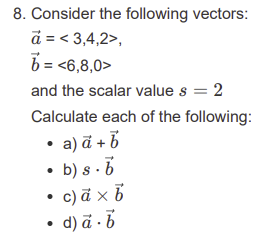
8. Consider the following vectors:

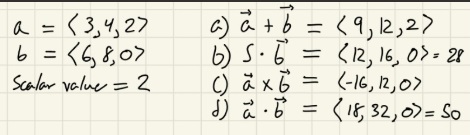
a = < 3,4,2>,

b = <6,8,0>

and the scalar value s = 2

Calculate each of the following:





9. Create a world transform matrix that translates by < 3,4,2> and rotates it 90° about the x-axis.

