**GELBEECK INFINITE**

**VIDEO PART 5**

**CONTINUOUS NODE MOVEMENT**

**IN PACMAN SCRIPT:**

private Vector2 NextDirection;*//Foreshadows to pacman where we want to move next when he reaches an intersection.*  
  
private Node CurrentNode; *//stores PacMan's current NodePosition.*  
private Node targetNode;*//PacMan's next Node.*  
private Node previousNode; *//Keep Track of where we came from.*

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

void ChangePosition(Vector2 d){  
  
        if(d!=direction)*//If there is a change in direction*  
            NextDirection = d;  
  
        if(CurrentNode != null){  
            Node moveToNode = CanMove(d); *//Get the next Node in that direction.*  
  
            if(moveToNode != null){  
                direction = d;  
                targetNode = MoveToNode; *//Next Node*  
                previousNode = CurrentNode; *//Current Node becomes previous node*  
                CurrentNode = null; *//current node becomes null because as we move we are not on a node.*  
  
            }  
              
        }  
              
    }

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**In start() method:** direction = Vector2.left;

**Because pac man always faces to the left before beginning.**

**Then:** ChangePosition(direction);

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

bool OverShotTarget(){  
        float nodeToTarget = LengthFromNode(targetNode.transform.position);  
        float nodeToSelf= LengthFromNode(transform.localPosition);  
  
        return nodeToSelf > nodeToTarget; *//Comparing the distance between Gelbeeck and the previous node*  
        *//versus the previous Node and the next Node. If Gelbeeck is larger that means we have overshot and*  
        *//returns true.*  
  
        }  
  
    float LengthFromNode(Vector2 targetPostion){  
        Vector2 vec = targetPosition - (Vector2)previousNode.transform.position;  
        return vec.sqrMagnitude;  
  
        }

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

    void Move(){  
        if(targetNode != CurrentNode && targetNode != null){  
  
            if(OverShotTarget()){

CurrentNode = targetNode; *//since we overshot our target*

                transform.localPosition = CurrentNode.transform.position;  
  
                Node moveToNode = CanMove(NextDirection); *//look for available nodes to move to in this direction.*  
  
                if(moveToNode != null)  
                    direction = NextDirection;    
  
                if(moveToNode == null)  
                    moveToNode = CanMove (direction); *//If we can't find next direction, find current direction Node*  
  
                if(moveToNode != null){  
  
                    targetNode = moveToNode;  
                    previousNode = CurrentNode;  
                    CurrentNode = null;  
  
                } else {  
                    direction = Vector2.zero; *//If there isn't any available direction. Stop.*

                }  
            }  
            else{  
                transform.localPosition+=(Vector3)direction\*speed\*Time.deltaTime; *//If we haven't Overshot, continue moving.*  
            }  
  
            }  
        }

**EDIT CHECKINPUT METHOD:**

void CheckInput(){  
        if(Input.GetKeyDown(KeyCode.LeftArrow)){  
            ChangePosition(Vector2.left);  
        }  
        else if (Input.GetKeyDown(KeyCode.RightArrow)){  
            ChangePosition(Vector2.right);  
        }  
        else if (Input.GetKeyDown(KeyCode.DownArrow)){  
            ChangePosition(Vector2.down);  
        }  
        else if(Input.GetKeyDown(KeyCode.UpArrow)){  
            ChangePosition(Vector2.up);  
        }

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**SETTING AN IDLE SPRITE:**

public Sprite idleSprite; **//DROP A SPRITE INTO THIS FIELD IN THE INSPECTOR.**

**IN UPDATE METHOD:**

UpdateAnimationState();

**CREATE METHOD:**

void UpdateAnimationState(){  
        if(direction == Vector2.zero){*//if not moving.*  
  
            GetComponent<Animator>().enabled = false; *//Disable animator*  
            GetComponent<SpriteRenderer>().sprite = idleSprite; *//set sprite to idleSprite*  
  
        }else{  
            GetComponent<Animator>().enabled = true; *//Enable animator*  
        }  
    }