**LAMPIRAN B**

**KODE IMPLEMENTASI GAME SPLATTED**

Segmen Program B.1 Class Set Objects

1. public class SetObjects : MonoBehaviour{
2. static int width;
3. static int height;
4. static int[,] stageFolded;
5. static int[,] stageUnfolded;
6. //0 = Kosong, 1 = batu, 2 = powerup, 3 = character, 4 = snowball (khusus unfolded)
7. Tilemap mapTilemap;
8. public TileBase rok;
9. public GameObject powerUpContainer;
10. public GameObject powerUp;
11. [SerializeField] GameObject playerManagerReference;
12. [SerializeField] bool only1BotActive;
13. [SerializeField] ColorManager colManager;
14. public static void initializeSize(int w, int h){
15. height = h;
16. width = w;
17. if (w % 2 == 1)
18. width--;
19. }
20. public static void setMap(int[,] stagearray, bool isFolded){
21. //mapFolded itu asumsikan di kiri
22. if (isFolded){
23. stageFolded = stagearray;
24. stageUnfolded = new int[height - 2, width - 2];
25. for (int i = 0; i < height - 2; i++){
26. for (int j = 0; j < (int)((width - 2) / 2); j++){
27. stageUnfolded[i, j] = stageFolded[i, j];
28. stageUnfolded[i, width - 3 - j] = stageFolded[i, j];
29. }
30. }
31. }
32. else{
33. stageUnfolded = stagearray;
34. height = stagearray.GetLength(0) + 2;
35. width = stagearray.GetLength(1) + 2;
36. }
37. }
38. public static void setMap(int index1, int index2, int number){
39. stageUnfolded[index1, index2] = number;
40. }
41. public static int getWidth(){
42. return width - 2;
43. }
44. public static int getHeight(){
45. return height - 2;
46. }

Segmen Program B.1 (Lanjutan)

1. public static int[,] getMap(bool folded){
2. if (folded)
3. return stageFolded;
4. else
5. return stageUnfolded;
6. }
7. // Start is called before the first frame update
8. void Start(){
9. mapTilemap = this.GetComponent<Tilemap>();
10. fillMap();
11. }
12. public void fillMap(){
13. //horizontal
14. for (int i = 0; i < width; i++){
15. mapTilemap.SetTile(new Vector3Int(i, 0, 1), rok);
16. mapTilemap.SetTile(new Vector3Int(i, -height + 1, 1), rok);
17. }
18. //vertikal
19. for (int i = 1; i < height - 1; i++){
20. mapTilemap.SetTile(new Vector3Int(0, -i, 1), rok);
21. mapTilemap.SetTile(new Vector3Int(width - 1, -i, 1), rok);
22. }
23. Coordinate tempCoor;
24. GameObject temp;
25. bool playerMade = false, oneBotAI = false;
26. for (int i = 0; i < height - 2; i++)
27. for (int j = 0; j < width - 2; j++){
28. tempCoor = new Coordinate(j, i);
29. if (stageUnfolded[i, j] == 1)
30. mapTilemap.SetTile(new Vector3Int(j + 1, -i - 1, 1), rok);
31. else if (stageUnfolded[i, j] == 2){
32. temp = Instantiate(powerUp, tempCoor.returnAsVector(), Quaternion.identity);
33. temp.transform.SetParent(powerUpContainer.transform);
34. }
35. else if (stageUnfolded[i, j] == 3){
36. if (!playerMade){
37. playerManagerReference.GetComponent <PlayersManager>().makeNewPlayer(tempCoor);
38. playerMade = true;
39. }
40. else if (only1BotActive && !oneBotAI){
41. playerManagerReference.GetComponent <PlayersManager>().makeNewBot(tempCoor,false);
42. oneBotAI = true;
43. }
44. else if(!only1BotActive)
45. playerManagerReference.GetComponent<PlayersManager> ().makeNewBot(tempCoor, j < (int)(width / 2) + 1);
46. }
47. }

Segmen Program B.1 (Lanjutan)

1. }
2. public void clearMap(){
3. mapTilemap.ClearAllTiles();
4. }
5. }

Segmen Program B.2 Class Players Manager

1. public class PlayersManager : MonoBehaviour{
2. GameObject[] players;
3. [SerializeField] Material material;
4. [SerializeField] GameObject playerPrefab;
5. [SerializeField] bool isAIActive;
6. [SerializeField] GameObject enemyPrefab;
7. [SerializeField] bool spawnPlayer;
8. [SerializeField] GameObject playersContainer;
9. [SerializeField] GameObject levelCamera;
10. List<bool[,]> accesibleAreas;
11. List<int[]> areaCorners;
12. private void Start(){
13. players = new GameObject[10];
14. foreach (Transform item in playersContainer.transform){
15. players[getFirstNullPlayerIndex()] = item.gameObject;
16. }
17. if (SetObjects.getMap(false) != null){
18. Queue<Coordinate> q = new Queue<Coordinate>();
19. Coordinate c, tempCoor;
20. accesibleAreas = new List<bool[,]>();
21. areaCorners = new List<int[]>();
22. int[,] currmap = SetObjects.getMap(false);
23. int[] coors;
24. bool[,] isChecked = new bool[currmap.GetLength(0), currmap.GetLength(1)], map;
25. for (int i = 0; i < currmap.GetLength(0); i++){
26. for (int j = 0; j < currmap.GetLength(1); j++){
27. if (!isChecked[i, j] && currmap[i, j] != 1){
28. coors = new int[4] { j, i, -1, -1 }; // x1,y1 (kiri atas),x2,y2 (kanan bawah)
29. map = new bool[currmap.GetLength(0), currmap.GetLength(1)];
30. isChecked[i, j] = true;
31. map[i, j] = true;
32. q.Enqueue(new Coordinate(j, i));
33. while (q.Count > 0){
34. c = q.Dequeue();
35. if (c.xCoor < coors[0]) coors[0] = c.xCoor;
36. else if (c.xCoor >= coors[2]) coors[2] = c.xCoor;
37. Segmen p if (c.yCoor < coors[1]) coors[1] = c.yCoor;
38. else if (c.yCoor >= coors[3]) coors[3] = c.yCoor;
39. for (int k = 0; k < 4; k++){

Segmen Program B.2 (Lanjutan)

1. tempCoor = new Coordinate(c.xCoor + Mathf.RoundToInt(Mathf.Sin(k \* Mathf.PI / 2)), c.yCoor + Mathf.RoundToInt(Mathf.Cos(k \* Mathf.PI / 2)));
2. if (tempCoor.xCoor >= 0 && tempCoor.yCoor >= 0 && tempCoor.yCoor < SetObjects.getHeight() && tempCoor.xCoor < currmap.GetLength(1) && currmap[tempCoor.yCoor, tempCoor.xCoor] != 1 && !isChecked[tempCoor.yCoor, tempCoor.xCoor]){
3. isChecked[tempCoor.yCoor, tempCoor.xCoor] = true;
4. map[tempCoor.yCoor, tempCoor.xCoor] = true;
5. q.Enqueue(tempCoor);
6. }
7. }
8. }
9. accesibleAreas.Add(map);
10. areaCorners.Add(coors);
11. }
12. }
13. }
14. Debug.Log(accesibleAreas.Count);
15. }
16. }
17. public void makeNewPlayer(Coordinate c){
18. int i = getFirstNullPlayerIndex();
19. players[i] = Instantiate(playerPrefab, c.returnAsVector(), Quaternion.identity);
20. levelCamera.GetComponent<CameraController2D>(). setCameraFollower(players[i], false);
21. players[i].transform.SetParent (playersContainer.transform);
22. }
23. public void makeNewBot(Coordinate c, bool isPlayerTeam){
24. int i = getFirstNullPlayerIndex();
25. GameObject tempEnemyPrefab = Instantiate(enemyPrefab, c.returnAsVector(), Quaternion.identity);
26. if (isPlayerTeam){
27. tempEnemyPrefab.GetComponent<SnowBrawler>().playerteam = true;
28. tempEnemyPrefab.GetComponent<ColorTaker>().id = 0;
29. }
30. for (int j = 0; j < accesibleAreas.Count; j++){
31. if (accesibleAreas[j][c.yCoor, c.xCoor]){
32. tempEnemyPrefab.GetComponent<BotActions>(). setMapSegmentID(j + 1, this);
33. break;
34. }
35. }
36. if (!isAIActive)
37. tempEnemyPrefab.GetComponent<StateMachine>().enabled = false;
38. players[i] = tempEnemyPrefab;

Segmen Program B.2 (Lanjutan)

1. players[i].transform.SetParent (playersContainer.transform);
2. }
3. public GameObject getnearestPlayer(Transform player, bool includeCollision, float visionRange){
4. float closestrange = 999, range;
5. int i = 0, index = -1;
6. RaycastHit2D cekKolisi;
7. bool isEnemy;
8. foreach (GameObject currplayer in players){
9. if (currplayer != null){
10. isEnemy = currplayer.GetComponent<SnowBrawler>().getplayerteam() != player.GetComponent<SnowBrawler>().getplayerteam();
11. Vector2 direction = Vector3.Normalize(currplayer.transform.position - player.transform.position);
12. cekKolisi = Physics2D.CircleCast(player.position, 0.4f, direction, visionRange, 64);
13. //Physics2D.CircleCast(currentPoint.returnAsVector(), circleSize, arah, dist, 64);
14. range = Vector2.Distance(player.transform.position, currplayer.transform.position);
15. if (range < closestrange && range > 0 && !(includeCollision && cekKolisi) && isEnemy){
16. closestrange = range;
17. index = i;
18. }
19. i++;
20. }
21. }
22. if (index >= 0)
23. return players[index];
24. else
25. return null;
26. }
27. public Coordinate getRandomSpot(int mapIndex){
28. bool[,] currAccesibleAreaa = accesibleAreas[mapIndex];
29. int[] currAreaCorners = areaCorners[mapIndex];
30. Coordinate tempCoor;
31. do{
32. tempCoor = new Coordinate(Random.Range(currAreaCorners[0], currAreaCorners[2] + 1), Random.Range(currAreaCorners[1], currAreaCorners[3] + 1));
33. } while (!currAccesibleAreaa[tempCoor.yCoor, tempCoor.xCoor]);
34. return tempCoor;
35. }
36. int getFirstNullPlayerIndex(){
37. for (int i = 0; i < players.Length; i++){
38. if (players[i] == null)

Segmen Program B.2 (Lanjutan)

1. return i;
2. }
3. return -1;
4. }
5. public void activatePlayersScript(bool activate){
6. MonoBehaviour[] scripts;
7. foreach (Transform item in playersContainer.transform){
8. scripts = item.GetComponents<MonoBehaviour>();
9. foreach (MonoBehaviour script in scripts){
10. script.enabled = activate;
11. }
12. item.GetComponent<ColorTaker>().enabled = true;
13. if (item.GetComponent<CoordinateMovement>() != null)
14. item.GetComponent<CoordinateMovement>().enabled = true;
15. }
16. }
17. public void gameOverAnimation(bool playerTeamWin){
18. foreach (Transform item in playersContainer.transform){
19. item.GetComponent<Animator>().SetBool("GameDone",true);
20. item.GetComponent<Animator>().SetBool("HasWon", item.GetComponent<SnowBrawler>().getplayerteam() == playerTeamWin);
21. }
22. }
23. }

Segmen Program B.3 Class Snow Ball Manager

1. public class SnowBallManager : MonoBehaviour{
2. public static SnowBallManager Instance;
3. public GameObject snowballscontainer;
4. [SerializeField] GameObject snowball;
5. [SerializeField] float respawnTime;
6. [SerializeField] int respawnAmount;
7. [SerializeField] ColorManager colManager;
8. float currentrespawnTimer;
9. private void Awake(){
10. if (Instance == null)
11. Instance = this;
12. }
13. // Start is called before the first frame update
14. void Start(){
15. currentrespawnTimer = respawnTime;
16. }
17. public void destroyball(int index){
18. GameObject ball = snowballscontainer.transform.GetChild(index).gameObject;
19. Coordinate ballcoor = AStarAlgorithm.vectorToCoordinate(ball.transform.position);
20. if(SetObjects.getMap(true) != null)

Segmen Program B.3 (Lanjutan)

1. SetObjects.setMap(ballcoor.yCoor, ballcoor.xCoor, 0);
2. Destroy(ball);
3. }
4. public void Update(){
5. if (currentrespawnTimer <= 0){
6. currentrespawnTimer = respawnTime;
7. putballs();
8. }
9. currentrespawnTimer -= Time.deltaTime;
10. }
11. void putballs(){
12. int x, y;
13. GameObject ballz;
14. for (int i = 0; i < respawnAmount; i++){
15. x = Mathf.RoundToInt(UnityEngine.Random.Range(0, SetObjects.getWidth() - 2));
16. y = Mathf.RoundToInt(UnityEngine.Random.Range(0, SetObjects.getHeight() - 2));
17. if (SetObjects.getMap(false)[y, x] == 0){
18. ballz = Instantiate(snowball, new Vector3(x + 1.5f, -y - 0.5f), Quaternion.identity);
19. ballz.transform.SetParent(snowballscontainer.transform ,true);
20. //Debug.Log("Bola ke-" + i + " = " + x + " " + y);
21. SetObjects.setMap(y, x, 4);
22. }
23. }
24. }
25. public void addBallinVector(Vector2 v){
26. GameObject ballz;
27. ballz = Instantiate(snowball,snowballscontainer.transform );
28. ballz.transform.position = v;
29. }
30. public bool deleteclosestball(Transform objecttransform, float rangetreshold){
31. bool isdeleted = false;
32. int index = getNearestBallIndex(objecttransform, rangetreshold);
33. if (index >= 0){
34. destroyball(index);
35. isdeleted = true;
36. }
37. return isdeleted;
38. }
39. public GameObject getClosestBall(Transform objecttransform, float rangetreshold){

Segmen Program B.3 (Lanjutan)

1. int index = getNearestBallIndex(objecttransform, rangetreshold);
2. return snowballscontainer.transform.GetChild(index).gameObject;
3. }
4. public int getNearestBallIndex(Transform objectTracked){
5. float closestrange = 999, range;
6. int i = 0, index = -1;
7. foreach (Transform ballz in snowballscontainer.transform){
8. range = Vector2.Distance(ballz.position, objectTracked.position);
9. if (range < closestrange && (ballz.GetComponent<PowerUp>() == null || ballz.GetComponent<PowerUp>().isActive())){
10. closestrange = range;
11. index = i;
12. }
13. i++;
14. }
15. return index;
16. }
17. public int getNearestBallIndex(Transform objectTracked, float range){
18. float closestrange = 999, currrange;
19. int i = 0, index = -1;
20. foreach (Transform ballz in snowballscontainer.transform){
21. currrange = Vector2.Distance(ballz.position, objectTracked.position);
22. if (currrange < range && currrange < closestrange && (ballz.GetComponent<PowerUp>() == null || ballz.GetComponent<PowerUp>().isActive())){
23. closestrange = currrange;
24. index = i;
25. }
26. i++;
27. }
28. if (index > -1 && Vector2.Distance(snowballscontainer.transform.GetChild(index).gameObject.transform.position, objectTracked.position) < range)
29. return index;
30. else
31. return -1;
32. }
33. public int getIndexfromSnowball(GameObject go){
34. int i = 0;
35. foreach (Transform item in snowballscontainer.transform){
36. if (item.gameObject == go)
37. return i;
38. i++;
39. }
40. return -1 ;
41. }

Segmen Program B.3 (Lanjutan)

1. public GameObject getBallfromIndex(int index){
2. try{
3. if (snowballscontainer.transform.childCount > 0)
4. return snowballscontainer.transform.GetChild(index).gameObject;
5. return null;
6. }
7. catch (System.Exception){
8. return null;
9. }
10. }
11. public bool isAnyBallNear(Vector2 position){
12. foreach (Transform item in snowballscontainer.transform){
13. if (Vector2.Distance(position, item.position) < 1 && (item.GetComponent<PowerUp>() == null || item.GetComponent<PowerUp>().isActive()))
14. return true;
15. }
16. return false;
17. }
18. public int getBallAmount(){
19. return snowballscontainer.transform.childCount;
20. }
21. }

Segmen Program B.4 Class BallMovement

1. public class BallMovement : MonoBehaviour{
2. [SerializeField] float speed;
3. [SerializeField] Vector2 direction;
4. [SerializeField] bool fromPlayerTeam;
5. [SerializeField] int ballScore;
6. Rigidbody2D thisRigid;
7. Collider2D currentCollider;
8. [SerializeField] int powerupId;
9. [SerializeField] GameObject thrower;
10. // Start is called before the first frame update
11. void Start(){
12. thisRigid = this.GetComponent<Rigidbody2D>();
13. }
14. public void initialize(float speed, Vector2 direction, bool isPlayerTeam, int ballScore, Collider2D you, GameObject thrower){
15. thisRigid = this.GetComponent<Rigidbody2D>();
16. this.speed = speed;
17. this.direction = direction;
18. this.fromPlayerTeam = isPlayerTeam;
19. this.ballScore = ballScore;
20. this.currentCollider = you;
21. if (you != null)

Segmen Program B.4 (Lanjutan)

1. Physics2D.IgnoreCollision(this.GetComponent< CircleCollider2D>(), you);
2. this.thrower = thrower;
3. thisRigid.rotation = - Vector2.SignedAngle(direction, Vector2.right);
4. }
5. public void initialize(float speed, Vector2 direction, bool isPlayerTeam, int ballScore, Collider2D you, GameObject thrower, int powerupId){
6. initialize(speed, direction, isPlayerTeam, ballScore, you, thrower);
7. this.powerupId = powerupId;
8. }
9. public void addScore(int score){
10. ballScore += score;
11. }
12. public void trySelfDestruct(GameObject collider){
13. if (powerupId == 0)
14. Destroy(gameObject);
15. else{
16. GetComponent<BallPowerUp>().modifyBall(collider);
17. }
18. }
19. private void FixedUpdate(){
20. thisRigid.MovePosition((Vector2)this.transform.position + direction \* Time.deltaTime \* speed);
21. if (powerupId != 2)
22. return;
23. Vector3 Rotation = new Vector3(0, 0, Time.deltaTime \* speed);
24. transform.Rotate(Rotation \* 25);
25. }
26. private void OnTriggerEnter2D(Collider2D collision){
27. AudioSource.PlayClipAtPoint(AudioScript.audioObject. getSound("Get"),transform.position);
28. if (collision.tag == "Wall"){
29. if (powerupId == 5 || powerupId == 3)
30. GetComponent<BallPowerUp>().modifyBall (collision.gameObject);
31. else
32. Destroy(gameObject);
33. }
34. }
35. public void ballIsCatched(bool isPlayerTeam, int addScore, float speed, Collider2D you, GameObject thrower){
36. this.fromPlayerTeam = isPlayerTeam;
37. this.ballScore += addScore;
38. this.speed += speed;
39. // Bisa kena pelempar sebelumnya
40. if (currentCollider != null)

Segmen Program B.4 (Lanjutan)

1. Physics2D.IgnoreCollision(this.GetComponent <CircleCollider2D>(), currentCollider, false);
2. //Ngga bisa kena pelempar baru
3. Physics2D.IgnoreCollision(this.GetComponent <CircleCollider2D>(), you);
4. currentCollider = you;
5. this.thrower = thrower;
6. }
7. public void setDirection(Vector2 direction){
8. this.direction = direction.normalized;
9. }
10. }

Segmen Program B.5 Class Ball Power Up

1. public class BallPowerUp : MonoBehaviour{
2. [SerializeField] int pierceScoreAdd;
3. [SerializeField] Sprite normalBallSprite;
4. [SerializeField] float explosionDelay;
5. [SerializeField] float explosionRadius;
6. [SerializeField] float movementSpeedSlow;
7. [SerializeField] float slowTime;
8. [SerializeField] int splitBalls;
9. [SerializeField] float splitRange;
10. float particleTimer;
11. BallMovement bmRef;
12. //Utk Powerup Sticky Bomb
13. Vector2 distance;
14. GameObject collision;
15. private void Awake(){
16. particleTimer = explosionDelay;
17. bmRef = GetComponent<BallMovement>();
18. }
19. public void modifyBall(GameObject collider){
20. SnowBrawler sbReff = collider.GetComponent<SnowBrawler>();
21. switch (bmRef.getBallPowerId()){
22. //Piercer
23. case 1:
24. bmRef.addScore(pierceScoreAdd);
25. break;
26. //Boomerang
27. case 2:
28. if (bmRef.getPlayerTeam() != sbReff.getplayerteam()){
29. bmRef.ballIsCatched(bmRef.getPlayerTeam(), sbReff.ballScoreAdd, sbReff.ballSpeedAdd, collider.GetComponent<BoxCollider2D>(),bmRef.getThrower());
30. bmRef.setDirection(bmRef.getThrower() .transform.position - transform.position);
31. }
32. else
33. Destroy(gameObject);
34. break;

Segmen Program B.5 (Lanjutan)

1. //Le Bombe
2. case 3:
3. GetComponent<CircleCollider2D>().enabled = false;
4. GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Static;
5. collision = collider;
6. distance = collider.transform.position - transform.position;
7. StartCoroutine(TimedExplode(explosionDelay));
8. break;
9. //Hu dingin
10. case 4:
11. sbReff.slowDown(movementSpeedSlow, slowTime);
12. Destroy(gameObject);
13. break;
14. //Tembok?
15. case 5:
16. Vector2 backPos = (Vector2)transform.position + bmRef.getDirection() \* 2f;
17. float angle = Vector2.SignedAngle((Vector2)transform.position, backPos);
18. float dist = Mathf.Sqrt(1 + Mathf.Pow(Mathf.Sin(Mathf.Deg2Rad \* angle \* 2), 2));
19. backPos = (Vector2)transform.position + bmRef.getDirection() \* dist \* 1.1f;
20. Collider2D[] explosiveCollision = Physics2D.OverlapCircleAll(backPos, 0.1f, 64);
21. if (explosiveCollision.Length == 0){
22. GameObject[] balls = new GameObject[splitBalls];
23. float initialAngle = -splitRange / 2;
24. for (int i = 0; i < splitBalls; i++){
25. balls[i] = Instantiate(gameObject, backPos, Quaternion.identity);
26. balls[i].GetComponent<BallMovement>(). setDirection(Quaternion.Euler(0, 0, initialAngle + i \* (splitRange / (splitBalls - 1))) \* bmRef.getDirection());
27. balls[i].GetComponent<BallMovement>(). setBallScore(Mathf.CeilToInt(bmRef.getBallScore() / 2));
28. balls[i].GetComponent<BallMovement>(). setPowerUpID(0);
29. balls[i].GetComponent<SpriteRenderer>().sprite = normalBallSprite;
30. balls[i].transform.localScale = new Vector3(0.7f, 0.7f, 1);
31. for (int j = 0; j < i; j++)
32. Physics2D.IgnoreCollision(balls[i].GetComponent <CircleCollider2D>(), balls[j].GetComponent<CircleCollider2D>());
33. }
34. }
35. Destroy(gameObject);
36. break;
37. }
38. }

Segmen Program B.5 (Lanjutan)

1. private void Update(){
2. if (GetComponent<BallMovement>().getBallPowerId() == 3 && particleTimer <= 0.3f)
3. transform.GetChild(0).gameObject.SetActive(true);
4. }
5. private void FixedUpdate(){
6. //biar bom bisa lekat ke target
7. if (bmRef.getBallPowerId() == 3 && collision != null){
8. particleTimer -= Time.deltaTime;
9. transform.position = (Vector2)collision.transform.position - distance;
10. }
11. }
12. IEnumerator TimedExplode(float seconds){
13. yield return new WaitForSeconds(seconds);
14. Collider2D[] explosiveCollision = Physics2D.OverlapCircleAll(transform.position, explosionRadius, 8);
15. int hitPlayers = 0;
16. Debug.Log("Boom");
17. foreach (Collider2D coll in explosiveCollision){
18. // Nggak isa melakukan coroutine kalau object ilang
19. coll.GetComponent<SnowBrawler>().getHit(0.5f,gameObject);
20. if (coll.GetComponent<SnowBrawler>().getplayerteam() != bmRef.getPlayerTeam())
21. hitPlayers++;
22. }
23. BarScoreManager.addscore(bmRef.getPlayerTeam(), bmRef.getBallScore() \* hitPlayers);
24. Destroy(gameObject);
25. }
26. }

Segmen Program B.6 Class SnowBrawler

1. public class SnowBrawler : MonoBehaviour{
2. protected int ballAmount { get; set; }
3. protected GameObject caughtBall { get; set; }
4. protected int ballPowerId { get; set; }
5. [SerializeField] GameObject displayedBall;
6. [SerializeField] GameObject numberReference;
7. public bool playerteam;
8. public float throwSpeed;
9. public float originalRunSpeed;
10. public float runSpeed;
11. public int ballScoreInitial;
12. public int ballScoreAdd;
13. public float ballSpeedAdd;
14. public float ballCatchTimer;
15. public float ballTakeRange;

Segmen Program B.6 (Lanjutan)

1. public GameObject ball;
2. public bool isAiming;
3. public float catchRecharge;
4. bool iscatching;
5. Sprite ballSprite;
6. Animator animator;
7. AudioSource SFXSource;
8. public bool canAct;
9. public bool canCatchBall;
10. public bool isTargeted;
11. Vector2 lastpos;
12. public float timeDelay = 0.1f;
13. float currentTimeDelay = 0;
14. public void Start(){
15. animator = GetComponent<Animator>();
16. SFXSource = GetComponent<AudioSource>();
17. isAiming = false;
18. canAct = true;
19. runSpeed = originalRunSpeed;
20. canCatchBall = true;
21. isTargeted = false;
22. }
23. public void Update(){
24. currentTimeDelay -= Time.deltaTime;
25. //update posisi sebelumnya target untuk prediksi
26. if (currentTimeDelay <= 0){
27. animator.SetFloat("MoveSpeed", Vector2.Distance(lastpos, transform.position));
28. if((Vector2)transform.position - (Vector2)lastpos!= Vector2.zero)
29. animator.SetFloat("SeeDirection", Vector2.Angle(Vector2.up, (Vector2)transform.position - (Vector2)lastpos));
30. lastpos = transform.position;
31. currentTimeDelay = timeDelay;
32. }
33. animator.SetBool("IsAiming", isAiming);
34. }
35. private void OnTriggerEnter2D(Collider2D collision){
36. if (collision.gameObject.tag == "Projectile"){
37. if (iscatching){
38. if (caughtBall != null)
39. Destroy(caughtBall);
40. caughtBall = collision.gameObject;
41. caughtBall.SetActive(false);
42. caughtBall.GetComponent<BallMovement>(). ballIsCatched(getplayerteam(), ballScoreAdd, ballSpeedAdd, GetComponent<BoxCollider2D>(), gameObject);
43. updateHoldedBallVisuals(false);
44. }
45. else{

Segmen Program B.6 (Lanjutan)

1. BallMovement bol = collision.gameObject.GetComponent<BallMovement>();
2. if (bol.getPlayerTeam() != playerteam)
3. BarScoreManager.addscore(bol.getPlayerTeam(), bol.getBallScore());
4. StartCoroutine(getHitNumerator(0.5f, collision.gameObject));
5. bol.trySelfDestruct(gameObject);
6. }
7. }
8. }
9. public void getBall(){
10. int ballindex = SnowBallManager.Instance.getNearestBallIndex(transform);
11. if (ballindex < 0 || Vector2.Distance(transform.position, SnowBallManager.Instance.getBallfromIndex(ballindex).transform.position) > ballTakeRange)
12. return;
13. if (SnowBallManager.Instance.getBallfromIndex(ballindex). GetComponent<PowerUp>()){
14. (ballSprite, ballPowerId) = SnowBallManager.Instance.getBallfromIndex(SnowBallManager.Instance.getNearestBallIndex(transform)).GetComponent<PowerUp>().getPowerupId();
15. if (ballPowerId > 0){
16. displayedBall.GetComponent<SpriteRenderer>().sprite = ballSprite;
17. ballAmount = 1;
18. }
19. }
20. else{
21. int deletedIndex = SnowBallManager.Instance. getNearestBallIndex(transform, ballTakeRange);
22. if (deletedIndex >= 0){
23. ballPowerId = 0;
24. SnowBallManager.Instance.deleteclosestball(transform, ballTakeRange);
25. ballAmount = 1;
26. ballSprite = ball.GetComponent<SpriteRenderer>().sprite;
27. }
28. }
29. SFXSource.clip = AudioScript.audioObject.getSound("Get");
30. SFXSource.Play();
31. updateHoldedBallVisuals(false);
32. }
33. public void shootBall(Vector2 direction){
34. GameObject ballin;
35. if (caughtBall != null){
36. ballin = caughtBall;
37. ballin.GetComponent<BallMovement>(). setDirection(direction);
38. ballin.transform.position = this.transform.position;

Segmen Program B.6 (Lanjutan)

1. ballin.SetActive(true);
2. caughtBall = null;
3. }
4. else{
5. ballin = Instantiate(ball, (Vector2)this.transform.position, Quaternion.identity);
6. ballin.GetComponent<BallMovement>(). initialize(throwSpeed, direction, playerteam, ballScoreInitial, this.GetComponent<BoxCollider2D>(), gameObject, ballPowerId);
7. ballAmount--;
8. if (ballPowerId > 0)
9. ballin.GetComponent<SpriteRenderer>().sprite = ballSprite;
10. }
11. SFXSource.clip = AudioScript.audioObject.getSound("Yeet");
12. SFXSource.Play();
13. ballin.GetComponent<SpriteRenderer>().material = GetComponent<SpriteRenderer>().material;
14. updateHoldedBallVisuals(true);
15. }
16. public int getBallAmount(){
17. return ballAmount + ((caughtBall == null) ? 0 : 1);
18. }
19. public void slowDown(float movementSpeedSlow, float slowTime){
20. StartCoroutine(slowDownNumerator(movementSpeedSlow, slowTime));
21. }
22. IEnumerator slowDownNumerator(float slowPower, float seconds){
23. GetComponent<SpriteRenderer>().color = new Color(11 / 255, 211 / 255, 1);
24. runSpeed = originalRunSpeed \* slowPower;
25. yield return new WaitForSeconds(seconds);
26. GetComponent<SpriteRenderer>().color = Color.white;
27. runSpeed = originalRunSpeed;
28. }
29. public void getHit(float seconds, GameObject snowBall){
30. StartCoroutine(getHitNumerator(seconds,snowBall));
31. }
32. public IEnumerator getHitNumerator(float seconds,GameObject snowBall){
33. if (snowBall.GetComponent<BallMovement>().getPlayerTeam() != playerteam){
34. GameObject numbers = Instantiate(numberReference);
35. numbers.GetComponent<NumbersController>(). setGambar(snowBall.GetComponent<BallMovement>().getBallScore());
36. numbers.GetComponent<NumbersController>(). StartingPosition = transform.position;

Segmen Program B.6 (Lanjutan)

1. }
2. canAct = false;
3. animator.SetBool("IsHit", true);
4. Debug.Log("Kena Hit");
5. yield return new WaitForSeconds(seconds);
6. canAct = true;
7. animator.SetBool("IsHit", false);
8. Debug.Log("Selesai Kena Hit");
9. }
10. public IEnumerator catchBall(){
11. runSpeed = 0;
12. iscatching = true;
13. canCatchBall = false;
14. animator.SetBool("IsCatching", true);
15. yield return new WaitForSeconds(ballCatchTimer);
16. animator.SetBool("IsCatching", false);
17. runSpeed = originalRunSpeed;
18. iscatching = false;
19. StartCoroutine(catchRecharging());
20. }
21. public IEnumerator catchRecharging(){
22. GetComponent<SpriteRenderer>().color = new Color(0.5f, 0.5f, 0.5f);
23. yield return new WaitForSeconds(catchRecharge);
24. GetComponent<SpriteRenderer>().color = new Color(1, 1, 1);
25. canCatchBall = true;
26. }
27. public IEnumerator shartShooting(){
28. animator.SetBool("isShooting", true);
29. canAct = false;
30. runSpeed = 0;
31. yield return new WaitForSeconds((0.5f \* 5) / 6);
32. runSpeed = originalRunSpeed;
33. canAct = true;
34. updateHoldedBallVisuals(true);
35. animator.SetBool("isShooting", false);
36. }
37. public void tryCatch(){
38. if (!iscatching)
39. StartCoroutine(catchBall());
40. }
41. void updateHoldedBallVisuals(bool isThrown){
42. if (GetComponent<DisplayBall>() != null)
43. GetComponent<DisplayBall>().updateUI(isThrown);
44. if (caughtBall == null && ballAmount == 0){
45. displayedBall.SetActive(false);
46. return;
47. }
48. GetComponent<Animator>().enabled = false;
49. displayedBall.SetActive(true);

Segmen Program B.6 (Lanjutan)

1. GetComponent<Animator>().enabled = true;
2. if (caughtBall != null){
3. displayedBall.GetComponent<SpriteRenderer>().sprite = caughtBall.GetComponent<SpriteRenderer>().sprite;
4. return;
5. }
6. displayedBall.GetComponent<SpriteRenderer>().sprite = ballSprite;
7. }
8. }

Segmen Program B.7 Class Shoot Mechanic

1. public class ShootMechanic : SnowBrawler{
2. [SerializeField] GameObject line1;
3. [SerializeField] GameObject line2;
4. [SerializeField] float aimAngle;
5. [SerializeField] float aimTime;
6. public float aimMovementSpeedPerc;
7. public bool IsFaking { get { return \_isFaking; } }
8. bool \_isFaking;
9. private float currentaimangle;
10. private float currentAimTime;
11. // Start is called before the first frame update
12. void Start(){
13. playerteam = true;
14. currentAimTime = 0;
15. isAiming = false;
16. isFaking = false;
17. base.Start();
18. }
19. // Update is called once per frame
20. void Update(){
21. base.Update();
22. if (!canAct){
23. isAiming = false;
24. line1.SetActive(false);
25. line2.SetActive(false);
26. return;
27. }
28. if (Input.GetKeyDown(KeyCode.E)){
29. getBall();
30. return;
31. }
32. //Mulai ngeaim
33. if (Input.GetMouseButtonDown(0) && (ballAmount > 0 || caughtBall != null) && !PauseGame.isPaused){
34. currentAimTime = aimTime;
35. isAiming = true;
36. line1.SetActive(true);
37. line2.SetActive(true);

Segmen Program B.7 (Lanjutan)

1. return;
2. }
3. //Fakeout
4. if(Input.GetMouseButtonDown(1) && isAiming){
5. line1.SetActive(false);
6. line2.SetActive(false);
7. isAiming = false;
8. StartCoroutine(Fakeout());
9. return;
10. }
11. if (!Input.GetMouseButton(0) && isAiming){
12. Vector2 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
13. Vector2 direction = mousePos - (Vector2)this.transform.position;
14. direction = Quaternion.AngleAxis(Random.Range(-(currentaimangle / 2), currentaimangle / 2), Vector3.forward) \* direction.normalized;
15. shootBall(direction);
16. isAiming = false;
17. line1.SetActive(false);
18. line2.SetActive(false);
19. StartCoroutine(shartShooting());
20. return;
21. }
22. //Kalkulasi 2 garis bidikan
23. if (isAiming){
24. Vector3 throwDir = Vector3.Normalize((Vector2)(Camera.main.ScreenToWorldPoint(Input.mousePosition) - transform.position));
25. float x = throwDir.x;
26. GetComponent<Animator>().SetFloat("SeeDirection", Vector2.Angle(Vector2.up, throwDir));
27. transform.localScale = new Vector3(Mathf.RoundToInt(x / Mathf.Abs(x)), 1, 1);
28. if (transform.localScale.x == -1)
29. throwDir = Vector3.Reflect(throwDir, Vector3.right);
30. currentAimTime -= Time.deltaTime;
31. currentaimangle = (currentAimTime < 0) ? 0 : (currentAimTime / aimTime) \* aimAngle;
32. line1.GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
33. line1.GetComponent<LineRenderer>().SetPosition(1, Quaternion.Euler(0, 0, currentaimangle \* -1 / 2) \* throwDir \* 20);
34. line2.GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
35. line2.GetComponent<LineRenderer>().SetPosition(1, Quaternion.Euler(0, 0, currentaimangle / 2) \* throwDir \* 20);
36. }
37. }
38. IEnumerator Fakeout(){
39. canAct = false;
40. isFaking = true;

Segmen Program B.7 (Lanjutan)

1. GetComponent<Animator>().SetBool("IsFaking",true);
2. yield return new WaitForSeconds(0.5f);
3. isFaking = false;
4. GetComponent<Animator>().SetBool("IsFaking", false);
5. isAiming = true;
6. line1.SetActive(true);
7. line2.SetActive(true);
8. canAct = true;
9. }
10. }

Segmen Program B.8 Class BotActions

1. public class BotActions : MonoBehaviour{
2. int mapSegmentid;
3. float sidewaysAngle;
4. bool debug;
5. Vector2 walkLocation;
6. float searchTimer, catchTimer;
7. [SerializeField] float aimSpeedPercentage;
8. //untuk vision dan nembak
9. [SerializeField] int linecastAmount;
10. [SerializeField] float linecastAngle;
11. [SerializeField] float castingLength;
12. [SerializeField] float sphereCastRadius;
13. [SerializeField] float catchTimerDelay;
14. [SerializeField] float catchChance;
15. GameObject sawBallGO, sawEnemyGO, sawProjectileGO;
16. [SerializeField] GameObject noticeMark;
17. Rigidbody2D thisRigid;
18. PlayersManager playerManagerRef;
19. GameObject target;
20. Vector2 lastpos, direction,viewDirection;
21. float timeDelay = 0.5f, currentTimeDelay = 0;
22. SnowBrawler snowBrawlerRef;
23. bool isSameBall;
24. private void Start(){
25. isSameBall = false;
26. searchTimer = 0;
27. catchTimer = 0;
28. thisRigid = GetComponent<Rigidbody2D>();
29. snowBrawlerRef = GetComponent<SnowBrawler>();
30. }
31. public void forgetTarget(){
32. target.GetComponent<SnowBrawler>().isTargeted = false;
33. target = null;
34. }
35. public void setTarget(GameObject target){
36. this.target = target;
37. target.GetComponent<SnowBrawler>().isTargeted = true;

Segmen Program B.8 (Lanjutan)

1. if (target.GetComponent<ShootMechanic>() != null)
2. StartCoroutine(visualiseNotice());
3. }
4. private void Update(){
5. searchTimer -= Time.deltaTime;
6. currentTimeDelay -= Time.deltaTime;
7. catchTimer -= Time.deltaTime;
8. //update posisi sebelumnya target untuk prediksi
9. if (target != null && currentTimeDelay <= 0){
10. lastpos = target.transform.position;
11. currentTimeDelay = timeDelay;
12. )
13. }
14. private void FixedUpdate(){
15. if (!snowBrawlerRef.canAct)
16. return;
17. sawBallGO = null; sawEnemyGO = null; sawProjectileGO = null;
18. RaycastHit2D currentHitObject;
19. float initialAngle = -linecastAngle / 2;
20. Vector2 currentDirection;
21. float shortestBallDist = 999, shortestEnemyDist = 999;
22. float currDistance;
23. for (int i = 0; i < linecastAmount; i++){
24. currentDirection = Quaternion.Euler(0, 0, initialAngle + i \* (linecastAngle / (linecastAmount - 1))) \* direction;
25. currentHitObject = Physics2D.Linecast((Vector2)transform.position + currentDirection / 2, (Vector2)transform.position + currentDirection \* castingLength);
26. //Kalau keliatan objek
27. if (currentHitObject){
28. currDistance = Vector2.Distance(transform.position, currentHitObject.collider.transform.position);
29. if (currentHitObject.collider.CompareTag("BallPile") && currDistance < shortestBallDist){
30. if (currentHitObject.collider.GetComponent<PowerUp>() == null || currentHitObject.collider.GetComponent<PowerUp>().isActive()){
31. shortestBallDist = currDistance;
32. sawBallGO = currentHitObject.collider.gameObject;
33. }
34. }
35. else if ((currentHitObject.collider.CompareTag("Player") || currentHitObject.collider.CompareTag("EnemyTeam")) && currDistance < shortestEnemyDist && !currentHitObject.collider.GetComponent<SnowBrawler>().isTargeted){

Segmen Program B.8 (Lanjutan)

1. if (currentHitObject.collider.GetComponent<SnowBrawler>().getplayerteam() != snowBrawlerRef.getplayerteam()){
2. shortestEnemyDist = currDistance;
3. sawEnemyGO = currentHitObject.collider.gameObject;
4. }
5. }
6. else if (currentHitObject.collider.CompareTag("Projectile")){
7. if (sawProjectileGO != currentHitObject.collider.gameObject)
8. isSameBall = false;
9. sawProjectileGO = currentHitObject.collider.gameObject;
10. }
11. }
12. }
13. if (searchTimer > 0 || !snowBrawlerRef.canAct)
14. return;
15. // koding sini lebih rapi ketimbang di Visual Script
16. if (!walkLocation.Equals(Vector2.zero)){
17. direction = Vector3.Normalize(walkLocation - (Vector2)transform.position);
18. viewDirection = direction;
19. thisRigid.MovePosition((Vector2)transform.position + (direction \* Time.deltaTime \* snowBrawlerRef.runSpeed \* (snowBrawlerRef.isAiming ? aimSpeedPercentage : 1)));
20. }
21. if (target != null)
22. viewDirection = Vector3.Normalize((Vector2)target.transform.position - (Vector2)transform.position);
23. if (viewDirection.x != 0)
24. transform.localScale = new Vector3(viewDirection.x / Mathf.Abs(viewDirection.x), transform.localScale.y, transform.localScale.z);
25. }
26. public void walkSideways(){
27. float angleOfChoice = ((Random.Range(0, 2) \* 2) - 1) \* 90;
28. angleOfChoice = angleOfChoice + Random.Range(-sidewaysAngle, sidewaysAngle + 1);
29. walkLocation = Quaternion.Euler(0, 0, angleOfChoice) \* (Vector2)(target.transform.position - transform.position);
30. }
31. public Vector2 GetAngle(GameObject them, float ballspeed){
32. //Diletakkan di lampiran berikutnya
33. }
34. public Coordinate[] getWaytoRandomCoordinate(){
35. Coordinate targetCoor;

Segmen Program B.8 (Lanjutan)

1. if (mapSegmentid > 0){
2. do{
3. targetCoor = playerManagerRef.getRandomSpot(mapSegmentid - 1);
4. } while (targetCoor.Equal(Coordinate.returnAsCoordinate(transform.position)));
5. //Debug.Log("Chosen " + (mapSegmentid - 1)+","+ target.ToString());
6. return AStarAlgorithm.makeWay(Coordinate.returnAsCoordinate(transform.position), targetCoor);
7. }
8. else{
9. do{
10. targetCoor = new Coordinate(Random.Range(0, SetObjects.getWidth() + 1), Random.Range(0, SetObjects.getHeight() + 1));
11. } while (targetCoor.Equal(Coordinate.returnAsCoordinate(transform.position)) || AStarAlgorithm.doAstarAlgo(Coordinate.returnAsCoordinate(transform.position), targetCoor, SetObjects.getMap(false)) == null);
12. }
13. return AStarAlgorithm.makeWay(Coordinate.returnAsCoordinate(transform.position), targetCoor);
14. }
15. public bool stillCanSeeTarget(){
16. Vector2 direction = target.transform.position - transform.position;
17. RaycastHit2D seenObject = Physics2D.Linecast((Vector2)transform.position + direction, target.transform.position,64);
18. return (!seenObject);
19. }
20. public bool isThrowBlocked(){
21. Vector2 direction = target.transform.position - transform.position;
22. //Hanya kalau diblok oleh batu
23. RaycastHit2D[] seenObject = Physics2D.CircleCastAll((Vector2)transform.position, 0.2f, direction, Vector2.Distance(target.transform.position, transform.position), 64);
24. foreach (RaycastHit2D item in seenObject){
25. if (item.collider.CompareTag("Wall"))
26. return true;
27. }
28. return false;
29. }
30. public void tryCatchBallChance(){

Segmen Program B.8 (Lanjutan)

1. if (Random.Range(1, 101) < catchChance && !isSameBall){
2. StartCoroutine(snowBrawlerRef.catchBall());
3. isSameBall = true;
4. }
5. catchTimer = catchTimerDelay;
6. Debug.Log("Coba Tangkap Bola");
7. }
8. public void tryCatchBall(){
9. if (catchTimer < 0){
10. StartCoroutine(snowBrawlerRef.catchBall());
11. catchTimer = catchTimerDelay;
12. Debug.Log("Coba Tangkap Bola");
13. }
14. }
15. public bool predictProjectileWillHit(){
16. if (sawProjectileGO == null)
17. return false;
18. Vector2 dir = Vector3.Normalize(transform.position - sawProjectileGO.transform.position);
19. RaycastHit2D[] objectsInWay = Physics2D.CircleCastAll(sawProjectileGO.transform.position, 0.15f, dir, Vector2.Distance(transform.position, sawProjectileGO.transform.position));
20. foreach (RaycastHit2D item in objectsInWay){
21. if (item.collider.CompareTag("Wall"))
22. return false;
23. if (item.collider.gameObject == gameObject && sawProjectileGO.GetComponent<BallMovement>().getThrower() != gameObject)
24. return true;
25. }
26. return false;
27. }
28. IEnumerator visualiseNotice(){
29. GetComponent<AudioSource>().clip = AudioScript.audioObject.getSound("Huhwhat");
30. GetComponent<AudioSource>().Play();
31. noticeMark.SetActive(true);
32. noticeMark.GetComponent<Animator>().Play("Base Layer.BotNotice");
33. yield return new WaitForSeconds(0.5f);
34. noticeMark.SetActive(false);
35. }
36. }

Segmen Program B.9 Fungsi GetAngle

1. public Vector2 GetAngle(GameObject them, float ballspeed){
2. //HUKUM COSINEE!!!!!!!!!
3. Vector2 pos1 = lastpos;
4. Vector2 pos2 = them.transform.position;
5. float targetSpeed = Vector2.Distance(pos1, pos2) / (timeDelay - currentTimeDelay);

Segmen Program B.9 (Lanjutan)

1. float angleBallandTarget = Vector2.SignedAngle(Vector3.Normalize((Vector2)transform.position - pos1), Vector3.Normalize(pos2 - pos1));
2. float initTargetandBallDistance = Vector2.Distance(pos2, transform.position);
3. float r = targetSpeed / ballspeed;
4. //Ini dapat 2 posibilitas jarak antara asal bola menuju muka target
5. float a = 1 - Mathf.Pow(r, 2);
6. float b = 2 \* Mathf.Cos(angleBallandTarget \* Mathf.Deg2Rad) \* r;
7. float c = -Mathf.Pow(initTargetandBallDistance, 2);
8. double isiAkar = Mathf.Pow(b, 2) - 4 \* a \* c;
9. isiAkar = Mathf.Sqrt((float)isiAkar);
10. double prediksi1 = (-b + isiAkar) / (2 \* a);
11. double prediksi2 = (-b - isiAkar) / (2 \* a);
12. if (debug){
13. Debug.Log("Speed,Angle dan Jarak Awal : " + targetSpeed + "," + angleBallandTarget + "," + initTargetandBallDistance);
14. Debug.DrawLine((Vector2)transform.position, pos1, Color.red, 5);
15. Debug.DrawLine(pos2, pos1, Color.red, 5);
16. Debug.Log("abc : " + a + "," + b + "," + c);
17. Debug.Log("Prediksi Jarak 1 & 2 : " + prediksi1 + "," + prediksi2);
18. }
19. //Dari jarak diatas ambil yang jarake lebih pendek
20. double prediksiFinal;
21. if (double.IsNaN(prediksi1) && double.IsNaN(prediksi2))
22. return Vector3.Normalize(pos2 - (Vector2)transform.position);
23. else if (!double.IsNaN(prediksi1) && !double.IsNaN(prediksi2)){
24. if (prediksi1 > 0 && (prediksi2 < 0 || prediksi1 < prediksi2))
25. prediksiFinal = prediksi1;
26. else if (prediksi2 > 0 && (prediksi1 < 0 || prediksi2 < prediksi1))
27. prediksiFinal = prediksi2;
28. else
29. return Vector3.Normalize(pos2 - (Vector2)transform.position);
30. }
31. else{
32. if (double.IsNaN(prediksi1))
33. prediksiFinal = prediksi1;
34. else
35. prediksiFinal = prediksi2;
36. if (prediksiFinal < 0)
37. return Vector3.Normalize(pos2 - (Vector2)transform.position);
38. }

Segmen Program B.9 (Lanjutan)

1. // Dari jarak yang didapat diambil waktu
2. double time = prediksiFinal / ballspeed;
3. return Vector3.Normalize((float)time \* targetSpeed \* (Vector2)Vector3.Normalize(pos2 - pos1) + pos2 - (Vector2)transform.position);
4. }

Segmen Program B.10 Fungsi Coordinate Movement

1. public class CoordinateMovement : MonoBehaviour{
2. Coordinate \_targetCoordinate;
3. Coordinate[] \_pathCoordinates;
4. int \_currentPathIndex;
5. BotActions \_botActRef;
6. private void Start(){
7. \_botActRef = GetComponent<BotActions>();
8. }
9. public void setTargetCoordinate(Vector2 targetCoordinate){
10. setTargetCoordinate( Coordinate.returnAsCoordinate(targetCoordinate));
11. }
12. public void setTargetCoordinate(Coordinate targetCoordinate){
13. \_targetCoordinate = targetCoordinate;
14. if (targetCoordinate.Equal(Coordinate.returnAsCoordinate(transform.position)))
15. \_pathCoordinates = new Coordinate[] { new Coordinate(targetCoordinate.xCoor, targetCoordinate.yCoor) };
16. else
17. \_pathCoordinates = AStarAlgorithm.makeWay(Coordinate.returnAsCoordinate(transform.position), targetCoordinate);
18. \_currentPathIndex = 0;
19. \_botActRef.setWalkLocation (\_pathCoordinates[\_currentPathIndex]);
20. }
21. public void setPathCoordinates(Coordinate[] pathCoordinate){
22. \_pathCoordinates = pathCoordinate;
23. \_targetCoordinate = pathCoordinate[pathCoordinate.Length - 1];
24. \_currentPathIndex = 0;
25. \_botActRef.setWalkLocation( \_pathCoordinates[\_currentPathIndex]);
26. }
27. // Update is called once per frame
28. void Update(){
29. if (\_targetCoordinate == null || \_currentPathIndex == \_pathCoordinates.Length || !GetComponent<SnowBrawler>().canAct)
30. return;

Segmen Program B.10 (Lanjutan)

1. //Debug.Log(\_pathCoordinates[\_currentPathIndex]);
2. if (Vector2.Distance(transform.position, \_pathCoordinates[\_currentPathIndex].returnAsVector()) < 0.25 && \_currentPathIndex< \_pathCoordinates.Length-1){
3. \_currentPathIndex++;
4. \_botActRef.setWalkLocation (\_pathCoordinates[\_currentPathIndex]);
5. }
6. }
7. public void stopMoving(){
8. \_targetCoordinate = null;
9. \_botActRef.setWalkLocation(Vector2.zero);
10. }
11. public bool hasArrivedtoDestination(){
12. return (Vector2.Distance (\_targetCoordinate.returnAsVector(), transform.position) < 0.25);
13. }
14. }

Segmen Program B.11 A Star Algorithm

1. public class AStarAlgorithm : MonoBehaviour{
2. public static float circleSize = 1f;
3. public static Coordinate[] makeWay(Transform character, Transform ball){
4. return makeWay(Coordinate.returnAsCoordinate(character.position), Coordinate.returnAsCoordinate(ball.position));
5. }
6. public static Coordinate[] makeWay(Coordinate character, Coordinate ball){
7. // layermask bukan integer tapi biner ternyata
8. Vector2 arah = (ball.returnAsVector() - character.returnAsVector());
9. arah = arah.normalized;
10. float dist = Vector3.Distance(ball.returnAsVector(), character.returnAsVector());
11. if (!Physics2D.CircleCast(character.returnAsVector(), circleSize, arah, dist, 64)){
12. Coordinate ints = vectorToCoordinate(ball.returnAsVector());
13. //Debug.Log("Shortcut");
14. return new Coordinate[] { ints };
15. }
16. else{
17. AstarNode result = doAstarAlgo(character, ball, SetObjects.getMap(false));
18. if (result == null || result.parentNode == null){

Segmen Program B.11 (Lanjutan)

1. Debug.Log("Null : " + character.ToString() + " ke " + ball.ToString());
2. return null;
3. }
4. ArrayList coordinates = new ArrayList();
5. while (result.parentNode != null){
6. coordinates.Add(result.coordinate);
7. result = result.parentNode;
8. }
9. coordinates.Reverse();
10. return finalcoordinates;
11. }
12. }
13. static Coordinate[] optimizePath(Coordinate[] path){
14. List<Coordinate> resultAL = new List<Coordinate>();
15. Coordinate currentPoint = path[0],checkedPoint = path[0];
16. int pathIndex = 1;
17. Vector2 arah ;
18. float dist;
19. resultAL.Add(currentPoint);
20. while (pathIndex < path.Length - 1){
21. arah = (path[pathIndex].returnAsVector() - currentPoint.returnAsVector()).normalized;
22. dist = Vector2.Distance(path[pathIndex].returnAsVector(), currentPoint.returnAsVector());
23. if (Physics2D.CircleCast(currentPoint.returnAsVector(), circleSize, arah, dist, 64)){
24. resultAL.Add(path[pathIndex-1]);
25. currentPoint = checkedPoint;
26. }
27. checkedPoint = path[pathIndex];
28. pathIndex++;
29. }
30. resultAL.Add(path[path.Length - 1]);
31. return resultAL.ToArray();
32. }
33. //Fix bisa
34. public static AstarNode doAstarAlgo(Transform character, Transform ball){
35. Coordinate posisikarakter = vectorToCoordinate(character.position);
36. Coordinate posisibola = vectorToCoordinate(ball.position);
37. //inisialisasi
38. int[,] map = SetObjects.getMap(false);
39. return doAstarAlgo(posisikarakter, posisibola, map);
40. }
41. public static AstarNode doAstarAlgo(Coordinate posisikarakter, Coordinate posisibola, int[,] map){
42. bool[,] isChecked = new bool[map.GetLength(0), map.GetLength(1)];
43. ArrayList listNode = new ArrayList{

Segmen Program B.11 (Lanjutan)

1. new AstarNode(posisikarakter, 0, Coordinate.Distance(posisikarakter, posisibola), null)
2. };
3. AstarNode currentnode, tempnode;
4. float distance;
5. bool isput,inBounds;
6. int mapheight = map.GetLength(0), maplength = map.GetLength(1);
7. Coordinate newcoor;
8. //Debug.Log($"Mulai debug Astar dari {posisikarakter} ke {posisibola}");
9. while (listNode.Count > 0){
10. currentnode = (AstarNode)listNode[0];
11. isChecked[currentnode.coordinate.yCoor, currentnode.coordinate.xCoor] = true;
12. listNode.RemoveAt(0);
13. if (currentnode.coordinate.xCoor == posisibola.xCoor && currentnode.coordinate.yCoor == posisibola.yCoor){
14. return currentnode;
15. }
16. for (int i = 0; i < 4; i++){
17. newcoor = new Coordinate(currentnode.coordinate.xCoor + Mathf.RoundToInt(Mathf.Sin(i \* Mathf.PI / 2)), currentnode.coordinate.yCoor + Mathf.RoundToInt(Mathf.Cos(i \* Mathf.PI / 2)));
18. inBounds = newcoor.yCoor >= 0 && newcoor.yCoor < mapheight && newcoor.xCoor >= 0 && newcoor.xCoor < maplength;
19. if (inBounds && map[newcoor.yCoor, newcoor.xCoor] != 1 && !isChecked[newcoor.yCoor, newcoor.xCoor]){
20. distance = Coordinate.Distance(newcoor, posisibola);
21. tempnode = new AstarNode(newcoor, currentnode.g + 1, distance, currentnode);
22. isput = false;
23. for (int j = 0; j < listNode.Count; j++){
24. if (((AstarNode)listNode[j]).f >= tempnode.f){
25. isChecked[newcoor.yCoor, newcoor.xCoor] = true;
26. isput = true;
27. listNode.Insert(j, tempnode);
28. break;
29. }
30. }
31. if (!isput)
32. listNode.Add(tempnode);
33. }
34. }
35. }
36. return null;
37. }
38. public static Coordinate vectorToCoordinate(Vector2 vent){
39. return new Coordinate(Mathf.RoundToInt(vent.x - 1.5f), Mathf.RoundToInt(-vent.y - 0.5f));
40. }
41. }

Segmen Program B.12 Bar Score Manager

1. public class BarScoreManager : MonoBehaviour{
2. static protected Slider playerTeamBar;
3. static protected Slider enemyTeamBar;
4. static protected TextMeshProUGUI textLeft;
5. static protected TextMeshProUGUI textRight;
6. [SerializeField] int maxScore;
7. [SerializeField] TextMeshProUGUI timerText;
8. [SerializeField] float fightLength;
9. [SerializeField] GameObject itemToAnimate;
10. [SerializeField] GameObject transition;
11. [SerializeField] TextMeshProUGUI gameOverText;
12. [SerializeField] PlayersManager playerManagerRef;
13. public bool StartTimer = false;
14. // Start is called before the first frame update
15. void Start(){
16. playerTeamBar = transform.GetChild(1).GetComponent<Slider>();
17. textLeft = transform.GetChild(1).GetChild(2).GetComponent<TextMeshProUGUI>();
18. enemyTeamBar = transform.GetChild(2).GetComponent<Slider>();
19. textRight = transform.GetChild(2).GetChild(2).GetComponent<TextMeshProUGUI>();
20. playerTeamBar.maxValue = maxScore;
21. enemyTeamBar.maxValue = maxScore;
22. playerTeamBar.value = 0;
23. enemyTeamBar.value = 0;
24. textLeft.text = "0";
25. textRight.text = "0";
26. timerText.text = Mathf.CeilToInt(fightLength).ToString().PadLeft(3, '0');
27. }
28. private void Update(){
29. if (!StartTimer)
30. return;
31. fightLength -= Time.deltaTime;
32. timerText.text = Mathf.CeilToInt(fightLength).ToString().PadLeft(3, '0');
33. if ((fightLength <= 0 || playerTeamBar.value >= maxScore || enemyTeamBar.value >= maxScore) && itemToAnimate != null)
34. StartCoroutine(victoryAnimation());
35. }
36. public static void addscore(bool forPlayerTeam, float amount){
37. if (playerTeamBar == null)
38. return;
39. if (forPlayerTeam)
40. playerTeamBar.value += amount;
41. else
42. enemyTeamBar.value += amount;

Segmen Program B.12 (Lanjutan)

1. updateScoreText();
2. }
3. static void updateScoreText(){
4. textLeft.text = playerTeamBar.value.ToString();
5. textRight.text = enemyTeamBar.value.ToString();
6. }
7. IEnumerator victoryAnimation(){
8. StartTimer = false;
9. playerManagerRef.activatePlayersScript(false);
10. itemToAnimate?.SetActive(true);
11. if (fightLength <= 0)
12. gameOverText.text = "Time Up!";
13. else
14. gameOverText.text = "Game Set!";
15. yield return new WaitForSecondsRealtime(4);
16. if (playerTeamBar.value >= enemyTeamBar.value)
17. gameOverText.text = "Player Team wins";
18. else if (enemyTeamBar.value >= playerTeamBar.value)
19. gameOverText.text = "enemy team wins";
20. playerManagerRef.gameOverAnimation(playerTeamBar.value >= enemyTeamBar.value);
21. yield return new WaitForSecondsRealtime(2);
22. GetComponent<MainMenuNavigation>().changeSceneIndex(-1);
23. }
24. }