**LAMPIRAN B**

**KODE IMPLEMENTASI GAME SPLATTED**

Segmen Program 7.1 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. }
48. public static int[,] getMap(bool folded){
49. if (folded)
50. return stageFolded;
51. else
52. return stageUnfolded;
53. }
54. // Start is called before the first frame update
55. void Start(){
56. mapTilemap = this.GetComponent<Tilemap>();
57. fillMap();
58. }
59. public void fillMap(){
60. //horizontal
61. for (int i = 0; i < width; i++){
62. mapTilemap.SetTile(new Vector3Int(i, 0, 1), rok);
63. mapTilemap.SetTile(new Vector3Int(i, -height + 1, 1), rok);
64. }
65. //vertikal
66. for (int i = 1; i < height - 1; i++){
67. mapTilemap.SetTile(new Vector3Int(0, -i, 1), rok);
68. mapTilemap.SetTile(new Vector3Int(width - 1, -i, 1), rok);
69. }
70. Coordinate tempCoor;
71. GameObject temp;
72. bool playerMade = false, oneBotAI = false;
73. for (int i = 0; i < height - 2; i++)
74. for (int j = 0; j < width - 2; j++){
75. tempCoor = new Coordinate(j, i);
76. if (stageUnfolded[i, j] == 1)
77. mapTilemap.SetTile(new Vector3Int(j + 1, -i - 1, 1), rok);
78. else if (stageUnfolded[i, j] == 2){
79. temp = Instantiate(powerUp, tempCoor.returnAsVector(), Quaternion.identity);
80. temp.transform.SetParent(powerUpContainer.transform);
81. }
82. else if (stageUnfolded[i, j] == 3){
83. if (!playerMade){
84. playerManagerReference.GetComponent <PlayersManager>().makeNewPlayer(tempCoor);
85. playerMade = true;
86. }
87. else if (only1BotActive && !oneBotAI){
88. playerManagerReference.GetComponent <PlayersManager>().makeNewBot(tempCoor,false);
89. oneBotAI = true;
90. }
91. else if(!only1BotActive)
92. playerManagerReference.GetComponent<PlayersManager> ().makeNewBot(tempCoor, j < (int)(width / 2) + 1);
93. }
94. }
95. }
96. public void clearMap(){
97. mapTilemap.ClearAllTiles();
98. }
99. }

Segmen Program 7.3 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++){
40. tempCoor = new Coordinate(c.xCoor + Mathf.RoundToInt(Mathf.Sin(k \* Mathf.PI / 2)), c.yCoor + Mathf.RoundToInt(Mathf.Cos(k \* Mathf.PI / 2)));
41. 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]){
42. isChecked[tempCoor.yCoor, tempCoor.xCoor] = true;
43. map[tempCoor.yCoor, tempCoor.xCoor] = true;
44. q.Enqueue(tempCoor);
45. }
46. }
47. }
48. accesibleAreas.Add(map);
49. areaCorners.Add(coors);
50. }
51. }
52. }
53. Debug.Log(accesibleAreas.Count);
54. }
55. }
56. public void makeNewPlayer(Coordinate c){
57. int i = getFirstNullPlayerIndex();
58. players[i] = Instantiate(playerPrefab, c.returnAsVector(), Quaternion.identity);
59. levelCamera.GetComponent<CameraController2D>(). setCameraFollower(players[i], false);
60. players[i].transform.SetParent (playersContainer.transform);
61. }
62. public void makeNewBot(Coordinate c, bool isPlayerTeam){
63. int i = getFirstNullPlayerIndex();
64. GameObject tempEnemyPrefab = Instantiate(enemyPrefab, c.returnAsVector(), Quaternion.identity);
65. if (isPlayerTeam){
66. tempEnemyPrefab.GetComponent<SnowBrawler>().playerteam = true;
67. tempEnemyPrefab.GetComponent<ColorTaker>().id = 0;
68. }
69. for (int j = 0; j < accesibleAreas.Count; j++){
70. if (accesibleAreas[j][c.yCoor, c.xCoor]){
71. tempEnemyPrefab.GetComponent<BotActions>(). setMapSegmentID(j + 1, this);
72. break;
73. }
74. }
75. if (!isAIActive)
76. tempEnemyPrefab.GetComponent<StateMachine>().enabled = false;
77. players[i] = tempEnemyPrefab;
78. players[i].transform.SetParent (playersContainer.transform);
79. }
80. public GameObject getnearestPlayer(Transform player, bool includeCollision, float visionRange){
81. float closestrange = 999, range;
82. int i = 0, index = -1;
83. RaycastHit2D cekKolisi;
84. bool isEnemy;
85. foreach (GameObject currplayer in players){
86. if (currplayer != null){
87. isEnemy = currplayer.GetComponent<SnowBrawler>().getplayerteam() != player.GetComponent<SnowBrawler>().getplayerteam();
88. Vector2 direction = Vector3.Normalize(currplayer.transform.position - player.transform.position);
89. cekKolisi = Physics2D.CircleCast(player.position, 0.4f, direction, visionRange, 64);
90. //Physics2D.CircleCast(currentPoint.returnAsVector(), circleSize, arah, dist, 64);
91. range = Vector2.Distance(player.transform.position, currplayer.transform.position);
92. if (range < closestrange && range > 0 && !(includeCollision && cekKolisi) && isEnemy){
93. closestrange = range;
94. index = i;
95. }
96. i++;
97. }
98. }
99. if (index >= 0)
100. return players[index];
101. else
102. return null;
103. }
104. public Coordinate getRandomSpot(int mapIndex){
105. bool[,] currAccesibleAreaa = accesibleAreas[mapIndex];
106. int[] currAreaCorners = areaCorners[mapIndex];
107. Coordinate tempCoor;
108. do{
109. tempCoor = new Coordinate(Random.Range(currAreaCorners[0], currAreaCorners[2] + 1), Random.Range(currAreaCorners[1], currAreaCorners[3] + 1));
110. } while (!currAccesibleAreaa[tempCoor.yCoor, tempCoor.xCoor]);
111. return tempCoor;
112. }
113. int getFirstNullPlayerIndex(){
114. for (int i = 0; i < players.Length; i++){
115. if (players[i] == null)
116. return i;
117. }
118. return -1;
119. }
120. public void activatePlayersScript(bool activate){
121. MonoBehaviour[] scripts;
122. foreach (Transform item in playersContainer.transform){
123. scripts = item.GetComponents<MonoBehaviour>();
124. foreach (MonoBehaviour script in scripts){
125. script.enabled = activate;
126. }
127. item.GetComponent<ColorTaker>().enabled = true;
128. if (item.GetComponent<CoordinateMovement>() != null)
129. item.GetComponent<CoordinateMovement>().enabled = true;
130. }
131. }
132. public void gameOverAnimation(bool playerTeamWin){
133. foreach (Transform item in playersContainer.transform){
134. item.GetComponent<Animator>().SetBool("GameDone",true);
135. item.GetComponent<Animator>().SetBool("HasWon", item.GetComponent<SnowBrawler>().getplayerteam() == playerTeamWin);
136. }
137. }
138. }

Segmen Program 7.4 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)
21. SetObjects.setMap(ballcoor.yCoor, ballcoor.xCoor, 0);
22. Destroy(ball);
23. }
24. public void Update(){
25. if (currentrespawnTimer <= 0){
26. currentrespawnTimer = respawnTime;
27. putballs();
28. }
29. currentrespawnTimer -= Time.deltaTime;
30. }
31. void putballs(){
32. int x, y;
33. GameObject ballz;
34. for (int i = 0; i < respawnAmount; i++){
35. x = Mathf.RoundToInt(UnityEngine.Random.Range(0, SetObjects.getWidth() - 2));
36. y = Mathf.RoundToInt(UnityEngine.Random.Range(0, SetObjects.getHeight() - 2));
37. if (SetObjects.getMap(false)[y, x] == 0){
38. ballz = Instantiate(snowball, new Vector3(x + 1.5f, -y - 0.5f), Quaternion.identity);
39. ballz.transform.SetParent(snowballscontainer.transform ,true);
40. //Debug.Log("Bola ke-" + i + " = " + x + " " + y);
41. SetObjects.setMap(y, x, 4);
42. }
43. }
44. }
45. public void addBallinVector(Vector2 v){
46. GameObject ballz;
47. ballz = Instantiate(snowball,snowballscontainer.transform );
48. ballz.transform.position = v;
49. }
50. public bool deleteclosestball(Transform objecttransform, float rangetreshold){
51. bool isdeleted = false;
52. int index = getNearestBallIndex(objecttransform, rangetreshold);
53. if (index >= 0){
54. destroyball(index);
55. isdeleted = true;
56. }
57. return isdeleted;
58. }
59. public GameObject getClosestBall(Transform objecttransform, float rangetreshold){
60. int index = getNearestBallIndex(objecttransform, rangetreshold);
61. return snowballscontainer.transform.GetChild(index).gameObject;
62. }
63. public int getNearestBallIndex(Transform objectTracked){
64. float closestrange = 999, range;
65. int i = 0, index = -1;
66. foreach (Transform ballz in snowballscontainer.transform){
67. range = Vector2.Distance(ballz.position, objectTracked.position);
68. if (range < closestrange && (ballz.GetComponent<PowerUp>() == null || ballz.GetComponent<PowerUp>().isActive())){
69. closestrange = range;
70. index = i;
71. }
72. i++;
73. }
74. return index;
75. }
76. public int getNearestBallIndex(Transform objectTracked, float range){
77. float closestrange = 999, currrange;
78. int i = 0, index = -1;
79. foreach (Transform ballz in snowballscontainer.transform){
80. currrange = Vector2.Distance(ballz.position, objectTracked.position);
81. if (currrange < range && currrange < closestrange && (ballz.GetComponent<PowerUp>() == null || ballz.GetComponent<PowerUp>().isActive())){
82. closestrange = currrange;
83. index = i;
84. }
85. i++;
86. }
87. if (index > -1 && Vector2.Distance(snowballscontainer.transform.GetChild(index).gameObject.transform.position, objectTracked.position) < range)
88. return index;
89. else
90. return -1;
91. }
92. public int getIndexfromSnowball(GameObject go){
93. int i = 0;
94. foreach (Transform item in snowballscontainer.transform){
95. if (item.gameObject == go)
96. return i;
97. i++;
98. }
99. return -1 ;
100. }
101. public GameObject getBallfromIndex(int index){
102. try{
103. if (snowballscontainer.transform.childCount > 0)
104. return snowballscontainer.transform.GetChild(index).gameObject;
105. return null;
106. }
107. catch (System.Exception){
108. return null;
109. }
110. }
111. public bool isAnyBallNear(Vector2 position){
112. foreach (Transform item in snowballscontainer.transform){
113. if (Vector2.Distance(position, item.position) < 1 && (item.GetComponent<PowerUp>() == null || item.GetComponent<PowerUp>().isActive()))
114. return true;
115. }
116. return false;
117. }
118. public int getBallAmount(){
119. return snowballscontainer.transform.childCount;
120. }
121. }

Segmen Program 7.5 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)
22. Physics2D.IgnoreCollision(this.GetComponent< CircleCollider2D>(), you);
23. this.thrower = thrower;
24. thisRigid.rotation = - Vector2.SignedAngle(direction, Vector2.right);
25. }
26. public void initialize(float speed, Vector2 direction, bool isPlayerTeam, int ballScore, Collider2D you, GameObject thrower, int powerupId){
27. initialize(speed, direction, isPlayerTeam, ballScore, you, thrower);
28. this.powerupId = powerupId;
29. }
30. public void addScore(int score){
31. ballScore += score;
32. }
33. public void trySelfDestruct(GameObject collider){
34. if (powerupId == 0)
35. Destroy(gameObject);
36. else{
37. GetComponent<BallPowerUp>().modifyBall(collider);
38. }
39. }
40. private void FixedUpdate(){
41. thisRigid.MovePosition((Vector2)this.transform.position + direction \* Time.deltaTime \* speed);
42. if (powerupId != 2)
43. return;
44. Vector3 Rotation = new Vector3(0, 0, Time.deltaTime \* speed);
45. transform.Rotate(Rotation \* 25);
46. }
47. private void OnTriggerEnter2D(Collider2D collision){
48. AudioSource.PlayClipAtPoint(AudioScript.audioObject. getSound("Get"),transform.position);
49. if (collision.tag == "Wall"){
50. if (powerupId == 5 || powerupId == 3)
51. GetComponent<BallPowerUp>().modifyBall (collision.gameObject);
52. else
53. Destroy(gameObject);
54. }
55. }
56. public void ballIsCatched(bool isPlayerTeam, int addScore, float speed, Collider2D you, GameObject thrower){
57. this.fromPlayerTeam = isPlayerTeam;
58. this.ballScore += addScore;
59. this.speed += speed;
60. // Bisa kena pelempar sebelumnya
61. if (currentCollider != null)
62. Physics2D.IgnoreCollision(this.GetComponent <CircleCollider2D>(), currentCollider, false);
63. //Ngga bisa kena pelempar baru
64. Physics2D.IgnoreCollision(this.GetComponent <CircleCollider2D>(), you);
65. currentCollider = you;
66. this.thrower = thrower;
67. }
68. public void setDirection(Vector2 direction){
69. this.direction = direction.normalized;
70. }
71. }

Segmen Program 7.6 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;
35. //Le Bombe
36. case 3:
37. GetComponent<CircleCollider2D>().enabled = false;
38. GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Static;
39. collision = collider;
40. distance = collider.transform.position - transform.position;
41. StartCoroutine(TimedExplode(explosionDelay));
42. break;
43. //Hu dingin
44. case 4:
45. sbReff.slowDown(movementSpeedSlow, slowTime);
46. Destroy(gameObject);
47. break;
48. //Tembok?
49. case 5:
50. Vector2 backPos = (Vector2)transform.position + bmRef.getDirection() \* 2f;
51. float angle = Vector2.SignedAngle((Vector2)transform.position, backPos);
52. float dist = Mathf.Sqrt(1 + Mathf.Pow(Mathf.Sin(Mathf.Deg2Rad \* angle \* 2), 2));
53. backPos = (Vector2)transform.position + bmRef.getDirection() \* dist \* 1.1f;
54. Collider2D[] explosiveCollision = Physics2D.OverlapCircleAll(backPos, 0.1f, 64);
55. if (explosiveCollision.Length == 0){
56. GameObject[] balls = new GameObject[splitBalls];
57. float initialAngle = -splitRange / 2;
58. for (int i = 0; i < splitBalls; i++){
59. balls[i] = Instantiate(gameObject, backPos, Quaternion.identity);
60. balls[i].GetComponent<BallMovement>(). setDirection(Quaternion.Euler(0, 0, initialAngle + i \* (splitRange / (splitBalls - 1))) \* bmRef.getDirection());
61. balls[i].GetComponent<BallMovement>(). setBallScore(Mathf.CeilToInt(bmRef.getBallScore() / 2));
62. balls[i].GetComponent<BallMovement>(). setPowerUpID(0);
63. balls[i].GetComponent<SpriteRenderer>().sprite = normalBallSprite;
64. balls[i].transform.localScale = new Vector3(0.7f, 0.7f, 1);
65. for (int j = 0; j < i; j++)
66. Physics2D.IgnoreCollision(balls[i].GetComponent <CircleCollider2D>(), balls[j].GetComponent<CircleCollider2D>());
67. }
68. }
69. Destroy(gameObject);
70. break;
71. }
72. }
73. private void Update(){
74. if (GetComponent<BallMovement>().getBallPowerId() == 3 && particleTimer <= 0.3f)
75. transform.GetChild(0).gameObject.SetActive(true);
76. }
77. private void FixedUpdate(){
78. //biar bom bisa lekat ke target
79. if (bmRef.getBallPowerId() == 3 && collision != null){
80. particleTimer -= Time.deltaTime;
81. transform.position = (Vector2)collision.transform.position - distance;
82. }
83. }
84. IEnumerator TimedExplode(float seconds){
85. yield return new WaitForSeconds(seconds);
86. Collider2D[] explosiveCollision = Physics2D.OverlapCircleAll(transform.position, explosionRadius, 8);
87. int hitPlayers = 0;
88. Debug.Log("Boom");
89. foreach (Collider2D coll in explosiveCollision){
90. // Nggak isa melakukan coroutine kalau object ilang
91. coll.GetComponent<SnowBrawler>().getHit(0.5f,gameObject);
92. if (coll.GetComponent<SnowBrawler>().getplayerteam() != bmRef.getPlayerTeam())
93. hitPlayers++;
94. }
95. BarScoreManager.addscore(bmRef.getPlayerTeam(), bmRef.getBallScore() \* hitPlayers);
96. Destroy(gameObject);
97. }
98. }

Segmen Program 7.8 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;
16. public GameObject ball;
17. public bool isAiming;
18. public float catchRecharge;
19. bool iscatching;
20. Sprite ballSprite;
21. Animator animator;
22. AudioSource SFXSource;
23. public bool canAct;
24. public bool canCatchBall;
25. public bool isTargeted;
26. Vector2 lastpos;
27. public float timeDelay = 0.1f;
28. float currentTimeDelay = 0;
29. public void Start(){
30. animator = GetComponent<Animator>();
31. SFXSource = GetComponent<AudioSource>();
32. isAiming = false;
33. canAct = true;
34. runSpeed = originalRunSpeed;
35. canCatchBall = true;
36. isTargeted = false;
37. }
38. public void Update(){
39. currentTimeDelay -= Time.deltaTime;
40. //update posisi sebelumnya target untuk prediksi
41. if (currentTimeDelay <= 0){
42. animator.SetFloat("MoveSpeed", Vector2.Distance(lastpos, transform.position));
43. if((Vector2)transform.position - (Vector2)lastpos!= Vector2.zero)
44. animator.SetFloat("SeeDirection", Vector2.Angle(Vector2.up, (Vector2)transform.position - (Vector2)lastpos));
45. lastpos = transform.position;
46. currentTimeDelay = timeDelay;
47. }
48. animator.SetBool("IsAiming", isAiming);
49. }
50. private void OnTriggerEnter2D(Collider2D collision){
51. if (collision.gameObject.tag == "Projectile"){
52. if (iscatching){
53. if (caughtBall != null)
54. Destroy(caughtBall);
55. caughtBall = collision.gameObject;
56. caughtBall.SetActive(false);
57. caughtBall.GetComponent<BallMovement>(). ballIsCatched(getplayerteam(), ballScoreAdd, ballSpeedAdd, GetComponent<BoxCollider2D>(), gameObject);
58. updateHoldedBallVisuals(false);
59. }
60. else{
61. BallMovement bol = collision.gameObject.GetComponent<BallMovement>();
62. if (bol.getPlayerTeam() != playerteam)
63. BarScoreManager.addscore(bol.getPlayerTeam(), bol.getBallScore());
64. StartCoroutine(getHitNumerator(0.5f, collision.gameObject));
65. bol.trySelfDestruct(gameObject);
66. }
67. }
68. }
69. public void getBall(){
70. int ballindex = SnowBallManager.Instance.getNearestBallIndex(transform);
71. if (ballindex < 0 || Vector2.Distance(transform.position, SnowBallManager.Instance.getBallfromIndex(ballindex).transform.position) > ballTakeRange)
72. return;
73. if (SnowBallManager.Instance.getBallfromIndex(ballindex). GetComponent<PowerUp>()){
74. (ballSprite, ballPowerId) = SnowBallManager.Instance.getBallfromIndex(SnowBallManager.Instance.getNearestBallIndex(transform)).GetComponent<PowerUp>().getPowerupId();
75. if (ballPowerId > 0){
76. displayedBall.GetComponent<SpriteRenderer>().sprite = ballSprite;
77. ballAmount = 1;
78. }
79. }
80. else{
81. int deletedIndex = SnowBallManager.Instance. getNearestBallIndex(transform, ballTakeRange);
82. if (deletedIndex >= 0){
83. ballPowerId = 0;
84. SnowBallManager.Instance.deleteclosestball(transform, ballTakeRange);
85. ballAmount = 1;
86. ballSprite = ball.GetComponent<SpriteRenderer>().sprite;
87. }
88. }
89. SFXSource.clip = AudioScript.audioObject.getSound("Get");
90. SFXSource.Play();
91. updateHoldedBallVisuals(false);
92. }
93. public void shootBall(Vector2 direction){
94. GameObject ballin;
95. if (caughtBall != null){
96. ballin = caughtBall;
97. ballin.GetComponent<BallMovement>(). setDirection(direction);
98. ballin.transform.position = this.transform.position;
99. ballin.SetActive(true);
100. caughtBall = null;
101. }
102. else{
103. ballin = Instantiate(ball, (Vector2)this.transform.position, Quaternion.identity);
104. ballin.GetComponent<BallMovement>(). initialize(throwSpeed, direction, playerteam, ballScoreInitial, this.GetComponent<BoxCollider2D>(), gameObject, ballPowerId);
105. ballAmount--;
106. if (ballPowerId > 0)
107. ballin.GetComponent<SpriteRenderer>().sprite = ballSprite;
108. }
109. SFXSource.clip = AudioScript.audioObject.getSound("Yeet");
110. SFXSource.Play();
111. ballin.GetComponent<SpriteRenderer>().material = GetComponent<SpriteRenderer>().material;
112. updateHoldedBallVisuals(true);
113. }
114. public int getBallAmount(){
115. return ballAmount + ((caughtBall == null) ? 0 : 1);
116. }
117. public void slowDown(float movementSpeedSlow, float slowTime){
118. StartCoroutine(slowDownNumerator(movementSpeedSlow, slowTime));
119. }
120. IEnumerator slowDownNumerator(float slowPower, float seconds){
121. GetComponent<SpriteRenderer>().color = new Color(11 / 255, 211 / 255, 1);
122. runSpeed = originalRunSpeed \* slowPower;
123. yield return new WaitForSeconds(seconds);
124. GetComponent<SpriteRenderer>().color = Color.white;
125. runSpeed = originalRunSpeed;
126. }
127. public void getHit(float seconds, GameObject snowBall){
128. StartCoroutine(getHitNumerator(seconds,snowBall));
129. }
130. public IEnumerator getHitNumerator(float seconds,GameObject snowBall){
131. if (snowBall.GetComponent<BallMovement>().getPlayerTeam() != playerteam){
132. GameObject numbers = Instantiate(numberReference);
133. numbers.GetComponent<NumbersController>(). setGambar(snowBall.GetComponent<BallMovement>().getBallScore());
134. numbers.GetComponent<NumbersController>(). StartingPosition = transform.position;
135. }
136. canAct = false;
137. animator.SetBool("IsHit", true);
138. Debug.Log("Kena Hit");
139. yield return new WaitForSeconds(seconds);
140. canAct = true;
141. animator.SetBool("IsHit", false);
142. Debug.Log("Selesai Kena Hit");
143. }
144. public IEnumerator catchBall(){
145. runSpeed = 0;
146. iscatching = true;
147. canCatchBall = false;
148. animator.SetBool("IsCatching", true);
149. yield return new WaitForSeconds(ballCatchTimer);
150. animator.SetBool("IsCatching", false);
151. runSpeed = originalRunSpeed;
152. iscatching = false;
153. StartCoroutine(catchRecharging());
154. }
155. public IEnumerator catchRecharging(){
156. GetComponent<SpriteRenderer>().color = new Color(0.5f, 0.5f, 0.5f);
157. yield return new WaitForSeconds(catchRecharge);
158. GetComponent<SpriteRenderer>().color = new Color(1, 1, 1);
159. canCatchBall = true;
160. }
161. public IEnumerator shartShooting(){
162. animator.SetBool("isShooting", true);
163. canAct = false;
164. runSpeed = 0;
165. yield return new WaitForSeconds((0.5f \* 5) / 6);
166. runSpeed = originalRunSpeed;
167. canAct = true;
168. updateHoldedBallVisuals(true);
169. animator.SetBool("isShooting", false);
170. }
171. public void tryCatch(){
172. if (!iscatching)
173. StartCoroutine(catchBall());
174. }
175. void updateHoldedBallVisuals(bool isThrown){
176. if (GetComponent<DisplayBall>() != null)
177. GetComponent<DisplayBall>().updateUI(isThrown);
178. if (caughtBall == null && ballAmount == 0){
179. displayedBall.SetActive(false);
180. return;
181. }
182. GetComponent<Animator>().enabled = false;
183. displayedBall.SetActive(true);
184. GetComponent<Animator>().enabled = true;
185. if (caughtBall != null){
186. displayedBall.GetComponent<SpriteRenderer>().sprite = caughtBall.GetComponent<SpriteRenderer>().sprite;
187. return;
188. }
189. displayedBall.GetComponent<SpriteRenderer>().sprite = ballSprite;
190. }
191. }

Segmen Program 7.10 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);
38. return;
39. }
40. //Fakeout
41. if(Input.GetMouseButtonDown(1) && isAiming){
42. line1.SetActive(false);
43. line2.SetActive(false);
44. isAiming = false;
45. StartCoroutine(Fakeout());
46. return;
47. }
48. if (!Input.GetMouseButton(0) && isAiming){
49. Vector2 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
50. Vector2 direction = mousePos - (Vector2)this.transform.position;
51. direction = Quaternion.AngleAxis(Random.Range(-(currentaimangle / 2), currentaimangle / 2), Vector3.forward) \* direction.normalized;
52. shootBall(direction);
53. isAiming = false;
54. line1.SetActive(false);
55. line2.SetActive(false);
56. StartCoroutine(shartShooting());
57. return;
58. }
59. //Kalkulasi 2 garis bidikan
60. if (isAiming){
61. Vector3 throwDir = Vector3.Normalize((Vector2)(Camera.main.ScreenToWorldPoint(Input.mousePosition) - transform.position));
62. float x = throwDir.x;
63. GetComponent<Animator>().SetFloat("SeeDirection", Vector2.Angle(Vector2.up, throwDir));
64. transform.localScale = new Vector3(Mathf.RoundToInt(x / Mathf.Abs(x)), 1, 1);
65. if (transform.localScale.x == -1)
66. throwDir = Vector3.Reflect(throwDir, Vector3.right);
67. currentAimTime -= Time.deltaTime;
68. currentaimangle = (currentAimTime < 0) ? 0 : (currentAimTime / aimTime) \* aimAngle;
69. line1.GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
70. line1.GetComponent<LineRenderer>().SetPosition(1, Quaternion.Euler(0, 0, currentaimangle \* -1 / 2) \* throwDir \* 20);
71. line2.GetComponent<LineRenderer>().SetPosition(0, Vector3.zero);
72. line2.GetComponent<LineRenderer>().SetPosition(1, Quaternion.Euler(0, 0, currentaimangle / 2) \* throwDir \* 20);
73. }
74. }
75. IEnumerator Fakeout(){
76. canAct = false;
77. isFaking = true;
78. GetComponent<Animator>().SetBool("IsFaking",true);
79. yield return new WaitForSeconds(0.5f);
80. isFaking = false;
81. GetComponent<Animator>().SetBool("IsFaking", false);
82. isAiming = true;
83. line1.SetActive(true);
84. line2.SetActive(true);
85. canAct = true;
86. }
87. }

Segmen Program 7.12 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;
38. if (target.GetComponent<ShootMechanic>() != null)
39. StartCoroutine(visualiseNotice());
40. }
41. private void Update(){
42. searchTimer -= Time.deltaTime;
43. currentTimeDelay -= Time.deltaTime;
44. catchTimer -= Time.deltaTime;
45. //update posisi sebelumnya target untuk prediksi
46. if (target != null && currentTimeDelay <= 0){
47. lastpos = target.transform.position;
48. currentTimeDelay = timeDelay;
49. )
50. }
51. private void FixedUpdate(){
52. if (!snowBrawlerRef.canAct)
53. return;
54. sawBallGO = null; sawEnemyGO = null; sawProjectileGO = null;
55. RaycastHit2D currentHitObject;
56. float initialAngle = -linecastAngle / 2;
57. Vector2 currentDirection;
58. float shortestBallDist = 999, shortestEnemyDist = 999;
59. float currDistance;
60. for (int i = 0; i < linecastAmount; i++){
61. currentDirection = Quaternion.Euler(0, 0, initialAngle + i \* (linecastAngle / (linecastAmount - 1))) \* direction;
62. currentHitObject = Physics2D.Linecast((Vector2)transform.position + currentDirection / 2, (Vector2)transform.position + currentDirection \* castingLength);
63. //Kalau keliatan objek
64. if (currentHitObject){
65. currDistance = Vector2.Distance(transform.position, currentHitObject.collider.transform.position);
66. if (currentHitObject.collider.CompareTag("BallPile") && currDistance < shortestBallDist){
67. if (currentHitObject.collider.GetComponent<PowerUp>() == null || currentHitObject.collider.GetComponent<PowerUp>().isActive()){
68. shortestBallDist = currDistance;
69. sawBallGO = currentHitObject.collider.gameObject;
70. }
71. }
72. else if ((currentHitObject.collider.CompareTag("Player") || currentHitObject.collider.CompareTag("EnemyTeam")) && currDistance < shortestEnemyDist && !currentHitObject.collider.GetComponent<SnowBrawler>().isTargeted){
73. if (currentHitObject.collider.GetComponent<SnowBrawler>().getplayerteam() != snowBrawlerRef.getplayerteam()){
74. shortestEnemyDist = currDistance;
75. sawEnemyGO = currentHitObject.collider.gameObject;
76. }
77. }
78. else if (currentHitObject.collider.CompareTag("Projectile")){
79. if (sawProjectileGO != currentHitObject.collider.gameObject)
80. isSameBall = false;
81. sawProjectileGO = currentHitObject.collider.gameObject;
82. }
83. }
84. }
85. if (searchTimer > 0 || !snowBrawlerRef.canAct)
86. return;
87. // koding sini lebih rapi ketimbang di Visual Script
88. if (!walkLocation.Equals(Vector2.zero)){
89. direction = Vector3.Normalize(walkLocation - (Vector2)transform.position);
90. viewDirection = direction;
91. thisRigid.MovePosition((Vector2)transform.position + (direction \* Time.deltaTime \* snowBrawlerRef.runSpeed \* (snowBrawlerRef.isAiming ? aimSpeedPercentage : 1)));
92. }
93. if (target != null)
94. viewDirection = Vector3.Normalize((Vector2)target.transform.position - (Vector2)transform.position);
95. if (viewDirection.x != 0)
96. transform.localScale = new Vector3(viewDirection.x / Mathf.Abs(viewDirection.x), transform.localScale.y, transform.localScale.z);
97. }
98. public void walkSideways(){
99. float angleOfChoice = ((Random.Range(0, 2) \* 2) - 1) \* 90;
100. angleOfChoice = angleOfChoice + Random.Range(-sidewaysAngle, sidewaysAngle + 1);
101. walkLocation = Quaternion.Euler(0, 0, angleOfChoice) \* (Vector2)(target.transform.position - transform.position);
102. }
103. public Vector2 GetAngle(GameObject them, float ballspeed){
104. //Dijelaskan di subbab berikutnya
105. }
106. public Coordinate[] getWaytoRandomCoordinate(){
107. Coordinate targetCoor;
108. if (mapSegmentid > 0){
109. do{
110. targetCoor = playerManagerRef.getRandomSpot(mapSegmentid - 1);
111. } while (targetCoor.Equal(Coordinate.returnAsCoordinate(transform.position)));
112. //Debug.Log("Chosen " + (mapSegmentid - 1)+","+ target.ToString());
113. return AStarAlgorithm.makeWay(Coordinate.returnAsCoordinate(transform.position), targetCoor);
114. }
115. else{
116. do{
117. targetCoor = new Coordinate(Random.Range(0, SetObjects.getWidth() + 1), Random.Range(0, SetObjects.getHeight() + 1));
118. } while (targetCoor.Equal(Coordinate.returnAsCoordinate(transform.position)) || AStarAlgorithm.doAstarAlgo(Coordinate.returnAsCoordinate(transform.position), targetCoor, SetObjects.getMap(false)) == null);
119. }
120. return AStarAlgorithm.makeWay(Coordinate.returnAsCoordinate(transform.position), targetCoor);
121. }
122. public bool stillCanSeeTarget(){
123. Vector2 direction = target.transform.position - transform.position;
124. RaycastHit2D seenObject = Physics2D.Linecast((Vector2)transform.position + direction, target.transform.position,64);
125. return (!seenObject);
126. }
127. public bool isThrowBlocked(){
128. Vector2 direction = target.transform.position - transform.position;
129. //Hanya kalau diblok oleh batu
130. RaycastHit2D[] seenObject = Physics2D.CircleCastAll((Vector2)transform.position, 0.2f, direction, Vector2.Distance(target.transform.position, transform.position), 64);
131. foreach (RaycastHit2D item in seenObject){
132. if (item.collider.CompareTag("Wall"))
133. return true;
134. }
135. return false;
136. }
137. public void tryCatchBallChance(){
138. if (Random.Range(1, 101) < catchChance && !isSameBall){
139. StartCoroutine(snowBrawlerRef.catchBall());
140. isSameBall = true;
141. }
142. catchTimer = catchTimerDelay;
143. Debug.Log("Coba Tangkap Bola");
144. }
145. public void tryCatchBall(){
146. if (catchTimer < 0){
147. StartCoroutine(snowBrawlerRef.catchBall());
148. catchTimer = catchTimerDelay;
149. Debug.Log("Coba Tangkap Bola");
150. }
151. }
152. public bool predictProjectileWillHit(){
153. if (sawProjectileGO == null)
154. return false;
155. Vector2 dir = Vector3.Normalize(transform.position - sawProjectileGO.transform.position);
156. RaycastHit2D[] objectsInWay = Physics2D.CircleCastAll(sawProjectileGO.transform.position, 0.15f, dir, Vector2.Distance(transform.position, sawProjectileGO.transform.position));
157. foreach (RaycastHit2D item in objectsInWay){
158. if (item.collider.CompareTag("Wall"))
159. return false;
160. if (item.collider.gameObject == gameObject && sawProjectileGO.GetComponent<BallMovement>().getThrower() != gameObject)
161. return true;
162. }
163. return false;
164. }
165. IEnumerator visualiseNotice(){
166. GetComponent<AudioSource>().clip = AudioScript.audioObject.getSound("Huhwhat");
167. GetComponent<AudioSource>().Play();
168. noticeMark.SetActive(true);
169. noticeMark.GetComponent<Animator>().Play("Base Layer.BotNotice");
170. yield return new WaitForSeconds(0.5f);
171. noticeMark.SetActive(false);
172. }
173. }

Segmen Program 7.17 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){
19. Debug.Log("Null : " + character.ToString() + " ke " + ball.ToString());
20. return null;
21. }
22. ArrayList coordinates = new ArrayList();
23. while (result.parentNode != null){
24. coordinates.Add(result.coordinate);
25. result = result.parentNode;
26. }
27. coordinates.Reverse();
28. return finalcoordinates;
29. }
30. }
31. static Coordinate[] optimizePath(Coordinate[] path){
32. List<Coordinate> resultAL = new List<Coordinate>();
33. Coordinate currentPoint = path[0],checkedPoint = path[0];
34. int pathIndex = 1;
35. Vector2 arah ;
36. float dist;
37. resultAL.Add(currentPoint);
38. while (pathIndex < path.Length - 1){
39. arah = (path[pathIndex].returnAsVector() - currentPoint.returnAsVector()).normalized;
40. dist = Vector2.Distance(path[pathIndex].returnAsVector(), currentPoint.returnAsVector());
41. if (Physics2D.CircleCast(currentPoint.returnAsVector(), circleSize, arah, dist, 64)){
42. resultAL.Add(path[pathIndex-1]);
43. currentPoint = checkedPoint;
44. }
45. checkedPoint = path[pathIndex];
46. pathIndex++;
47. }
48. resultAL.Add(path[path.Length - 1]);
49. return resultAL.ToArray();
50. }
51. //Fix bisa
52. public static AstarNode doAstarAlgo(Transform character, Transform ball){
53. Coordinate posisikarakter = vectorToCoordinate(character.position);
54. Coordinate posisibola = vectorToCoordinate(ball.position);
55. //inisialisasi
56. int[,] map = SetObjects.getMap(false);
57. return doAstarAlgo(posisikarakter, posisibola, map);
58. }
59. public static AstarNode doAstarAlgo(Coordinate posisikarakter, Coordinate posisibola, int[,] map){
60. bool[,] isChecked = new bool[map.GetLength(0), map.GetLength(1)];
61. ArrayList listNode = new ArrayList{
62. new AstarNode(posisikarakter, 0, Coordinate.Distance(posisikarakter, posisibola), null)
63. };
64. AstarNode currentnode, tempnode;
65. float distance;
66. bool isput,inBounds;
67. int mapheight = map.GetLength(0), maplength = map.GetLength(1);
68. Coordinate newcoor;
69. //Debug.Log($"Mulai debug Astar dari {posisikarakter} ke {posisibola}");
70. while (listNode.Count > 0){
71. currentnode = (AstarNode)listNode[0];
72. isChecked[currentnode.coordinate.yCoor, currentnode.coordinate.xCoor] = true;
73. listNode.RemoveAt(0);
74. if (currentnode.coordinate.xCoor == posisibola.xCoor && currentnode.coordinate.yCoor == posisibola.yCoor){
75. return currentnode;
76. }
77. for (int i = 0; i < 4; i++){
78. 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)));
79. inBounds = newcoor.yCoor >= 0 && newcoor.yCoor < mapheight && newcoor.xCoor >= 0 && newcoor.xCoor < maplength;
80. if (inBounds && map[newcoor.yCoor, newcoor.xCoor] != 1 && !isChecked[newcoor.yCoor, newcoor.xCoor]){
81. distance = Coordinate.Distance(newcoor, posisibola);
82. tempnode = new AstarNode(newcoor, currentnode.g + 1, distance, currentnode);
83. isput = false;
84. for (int j = 0; j < listNode.Count; j++){
85. if (((AstarNode)listNode[j]).f >= tempnode.f){
86. isChecked[newcoor.yCoor, newcoor.xCoor] = true;
87. isput = true;
88. listNode.Insert(j, tempnode);
89. break;
90. }
91. }
92. if (!isput)
93. listNode.Add(tempnode);
94. }
95. }
96. }
97. return null;
98. }
99. public static Coordinate vectorToCoordinate(Vector2 vent){
100. return new Coordinate(Mathf.RoundToInt(vent.x - 1.5f), Mathf.RoundToInt(-vent.y - 0.5f));
101. }
102. }

Segmen Program 7.18 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;
43. updateScoreText();
44. }
45. static void updateScoreText(){
46. textLeft.text = playerTeamBar.value.ToString();
47. textRight.text = enemyTeamBar.value.ToString();
48. }
49. IEnumerator victoryAnimation(){
50. StartTimer = false;
51. playerManagerRef.activatePlayersScript(false);
52. itemToAnimate?.SetActive(true);
53. if (fightLength <= 0)
54. gameOverText.text = "Time Up!";
55. else
56. gameOverText.text = "Game Set!";
57. yield return new WaitForSecondsRealtime(4);
58. if (playerTeamBar.value >= enemyTeamBar.value)
59. gameOverText.text = "Player Team wins";
60. else if (enemyTeamBar.value >= playerTeamBar.value)
61. gameOverText.text = "enemy team wins";
62. playerManagerRef.gameOverAnimation(playerTeamBar.value >= enemyTeamBar.value);
63. yield return new WaitForSecondsRealtime(2);
64. GetComponent<MainMenuNavigation>().changeSceneIndex(-1);
65. }
66. }