BAB VII

IMPLEMENTASI GAME

Dengan level selesai diimplementasikan, sekarang kita bisa mempersiapkan komponen komponen dalam level yang membuat Splatted layak dipanggil sebuah game. Sama seperti bab 6, bab ini akan menjelaskan seluruh kode yang dipakai untuk komponen game dan penjelasan mengapa kode tersebut seperti berikut. Di bab ini hanya dijelaskan komponen yang digunakan dalam level dengan komponen – kompenen lain hanya disinggung atau dijelaskan sekilas agar bab ini tidak terlalu panjang.

1. Bola Salju

Dalam subbab ini akan dijelaskan aspek – aspek yang berhubungan dengan bola salju, dari cara memunculkan salju di arena hingga cara kerja bola spesial, dimulai dengan kompenen yang mengatur semua hal yang behubungan dengan bola salju : Snow Ball Manager.

1. Snowball Manager

Snowball Manager adalah sebuah class yang bertanggung jawab untu nge-spawn gumpalan - gumpalan salju di level dan juga mengurusi interaksi antara karakter game dengan gumpalan salju. Snowball manager menggunakan sebuah GameObject yang beraksi sebagai wadah para gumpalan salju agar bisa memudahkan akses ketimbang mencari seluruh GameObject di level.



Gambar 7.X

Gumpalan salju yang di-spawn secara berkala

Berikut adalah class dari SnowBallManager:

Segmen Program 7.X 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);

Segmen Program 7.X (Lanjutan)

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

Segmen Program 7.X (Lanjutan)

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

Sebelum memulai penjelasan dari potongan program diatas, perlu diketahui bila ini adalah salah satu segmen program pertama yang dikerjakan, sehingga kode di dalam segmen ini bisa terlihat lebih berantakan ketimbang segmen program lainnya. Dengan itu berikut ini adalah penjelasan untuk masing masing segmen program yang panjang diatas :

* 2 : Instance ini adalah sebuah static SnowballManager yang memperbolehkan setiap layar memiliki hanya 1 SnowballManager, dan memudahkan akses dari komponen game lain untuk mengakses snowballManager.
* 3 : GameObject yang akan menampung semua bola salju.
* 5 : Referensi GameObject gumpalan salju yang akan dispawn
* 6 : Periode sebuah gumpalan salju dipanggil.
* 7 : Menentukan berapa banyak gumpalan salju yang dispawn saat waktunya nge-spawn
* 8 : Referensi ke komponen Color Manager.
* 9 : Variabel untuk mencatat timer sebelum bola salju di-spawn
* 11 – 14 : Saat Gameobject nyala, maka akan dicek apakan Instance SnowBallManager kosong, bila kosong maka Instance akan diisi dengan SnowBallManager sekarang.
* 17 – 19 : Di frame pertama, currentRespawnTimer akan diisi dengan respawnTime, menandakan bila timer telah dimulai.
* 21 – 27 : Fungsi ini meminta parameter index dan bongkahan salju yang ke – index akan dihapus. Fungsi ini dipakai untuk mengambil sebuah bola, dimana mengambil bola dari bongkahan salju, bongkahan tersebut akan hilang.
* 29 – 35 : Setiap frame fungsi update ini akan dipanggil, dan seluruh isi update akan dijalankan.
* 30 – 33 : Bila timer sudah habis, maka reset timer dan taruh gumapalan salju.
* 34 : Kurangi waktu di timer sebanyak lama waktu sebelum frame terakhir.
* 38 – 51 : Fungsi untuk menaruh gumpalan salju yang digunakan di baris 32. Pada dasarnya fungsi ini menaruh salju sebanyak yang ditentukan variabel respawnAmount, dan setiap kali fungsi mau menaruh bola dicek apabila tile yang mau ditempati kosong. Bila kosong maka salju diletakkan di situ, tapi bila ada sesuatu, maka salju yang sekarang dilewati dan dilanjuti salju selanjutnya.
* 53 – 57 : Fungsi untuk meletakkan bola di sebuah koordinat.
* 59 – 68 : Fungsi yang dipakai untuk menghapus gumpalan salju yang terdekat. Pertama diambil gumpalan salju terdekat menggunakan fungsi di baris 74, lalu diperiksa apakah gumpalan tersebut cukup dekat dengan parameter objecttransform. Bila cukup dekat (ditentukan oleh parameter rangeThreshold), maka salju akan dihapus dan direturn true sebagai tanda bila pengambilan bola berhasil, tapi bila bola tidak ditemukan atau terlalu jauh, maka direturn false.
* 69 – 72 : Serupa dengan fungsi diatas, dimana ketimbang mengambil bola lalu menghapus, fungsi ini mengambil bola terdekat, dan bila fungsi mendapat gumpalan yang cukup dekat maka gumpalan tersebut akan direturn, tapi bila tidak maka tidak dikembalikan apa – apa.
* 74 – 86 : Fungsi ini mengambil bola salju terdekat dari parameter Transform objectTracked, caranya adalah dengan menghitung jarak dari objectTracked tersebut dan setiap gumpalan salju dalam GameObject snowballscontainer, diperiksa satu – satu, simpan index bola yang paling dekat dan return index tersebut.
* 88 – 103 : Fungsi ini merupakan copasan dari fungsi diatas , dengan perbedaan utama merupakan setelah mendapat index dari salju terdekat, diperiksa apakah salju tersebut cukup dekat dengan parameter range.
* 105 – 113 : Fungsi ini dipakai untuk mengambil sebuah index dari gumpalan salju dengan GameObject gumpalan tersebut sebagai parameter, diperiksa seluruh gumpalan salju di snowballscontainer dan direturn index salju yang sama. Bila tidak ditemukan maka dikembalikan index -1.
* 115 – 124 : Kebalikan dari atas, fungsi ini mereturn sebuah GameObject dari parameter index yang diberikan dari snowballscontainer.
* 126 – 132 : Fungsi yang digunakan untuk mengecek apabila ada salju di dekat parameter position, bila ada return true, bila tidak return false.
* 134 – 136 : Return jumlah salju yang sekarang ini ada di dalam level.

1. Ball Movement

Script selanjutnya adalah Ball Movement atau pergerakan bola, sebuah script yang membantu bola yang telah dilempar untuk tetap terbang. Script ini mengandung variabel untuk terbang seperti arah terbang dan kecepatan terbang, dan juga mengandung variabel – variabel lain yang penting seperti tim pelempar bola, skor bola bila mengenai musuh dll.

Berikut adalah Class tersebut :

Segmen Program 7.X 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. }

Segmen Program 7.X (Lanjutan)

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

Segmen Program 7.X (Lanjutan)

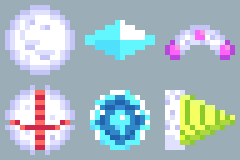
1. this.thrower = thrower;
2. }
3. public void setDirection(Vector2 direction){
4. this.direction = direction.normalized;
5. }
6. }

Di Class diatas seluruh Get & Set standar telah dihapus agar tidak memanjangkan buku, berikut penjelasannya :

* 3 : Variabel penentu kecepatan terbang bola.
* 4 : Variabel penentu arah terbang bola
* 5 : Variabel penunjuk apabila bola dilempar oleh tim player atau musuh.
* 6 : Skor yang akan diberikan bila bola mengenai lawan.
* 7 : Menyimpan referensi Rigidbody bola salju
* 8 : Menyimpan collider dari pelempar bola.
* 9 : Menyimpan id power up yang dimiliki oleh bola, bila id 0 maka bola merupakan bola biasa, tapi bila diatas 0 maka bola adalah bola spesial.
* 10 : Menyimpan GameObject pelempar bola salju
* 13 – 15 : Mengisi thisRigid dengan RigidBody2D dari bola salju.
* 17 – 28 : Fungsi ini digunakan untuk menginialisasi sebuah bola, dimana inisialisasi ini terjadi pada saat bola dilempar.
* 18 : Meskipun fungsi ini ada di Start, tidak bisa dipastikan yang mana yang jalan duluan jadi agar aman ditaruh disini juga
* 24 – 25 : Bila pelempar memiliki sebuah CircleCollider2D, maka bola akan mengabaikan pelempar tersebut, ini berguna agar bola tidak hilang saat dilempar karena bola mengenai diri sendiri.
* 27 : Menentukan rotasi dari bola menggunakan RigidBody2D.
* 30 – 33 : Fungsi ini sama dengan fungsi initialize sebelumnya, dengan perbedaan utama adalah fungsi ini juga menentukan id dari bola spesial, berguna saat player melempar bola spesial.
* 35 – 37 : Menambah skor yang bisa diberi bola salju, biasa dipanggil saat bola ditangkap
* 39 – 45 : Coba hancurkan bola. Bila bola mengenai sesuatu dan merupakan bola biasa, maka bola hancur, tapi bila bola merupakan bola spesial, maka bola akan melakukan aksi berdasarkan id power up yang dimiliki.
* 48 : Dalam interval fixed update yang dipanggil setiap frame sesuai dengan logika fisika unity, bola akan digerakkan sesuai dengan arah dan kecepatan yang sudah disimpan.
* 49 – 52 : Bila bola spesial adalah sebuah Snow-a-rang, maka bola akan berputar setelah bergerak.
* 55 – 63 : Bila bola mengenai sesuatu, maka mainkan suara yang dipersiapkan di tempat bola bertubrukan. Lalu bila bola mengenai dinding, maka diperiksa apabila bola merupakan Stone Auger (bor) atau Explod-o-ball (bom), maka aktifkan aksi dari bola tersebut, tapi bila bukan dua jenis tersebut, maka bola hancur.
* 65 – 76 : Fungsinya mirip dengan inisialisasi, dengan perbedaan utama ini dimodifikasi untuk dipakai saat seorang player menangkap sebuah bola.
* 78 – 80 : Fungsi ini dipakai untuk megganti arah terbang dari bola.

1. Bola Spesial

Dengan Class sebelumnya berfokus pada fitur – fitur bola biasa seperti pergerakan dan apa yang terjadi saat mengenai sesuatu, class Ball Power Up ini berfokus pada apa yang terjadi pada saat bola spesial mengenai sebuah target. Secara total terdapat 5 bola spesial, masing masing dengan efek khusus tersendiri.



Gambar 7.X

Bola – bola salju dengan power up id 0 - 5

Berikut adalah Class Ball Power Up :

Segmen Program 7.X 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:

Segmen Program 7.X (Lanjutan)

1. sbReff.slowDown(movementSpeedSlow, slowTime);
2. Destroy(gameObject);
3. break;
4. //Tembok?
5. case 5:
6. Vector2 backPos = (Vector2)transform.position + bmRef.getDirection() \* 2f;
7. float angle = Vector2.SignedAngle((Vector2)transform.position, backPos);
8. float dist = Mathf.Sqrt(1 + Mathf.Pow(Mathf.Sin(Mathf.Deg2Rad \* angle \* 2), 2));
9. backPos = (Vector2)transform.position + bmRef.getDirection() \* dist \* 1.1f;
10. Collider2D[] explosiveCollision = Physics2D.OverlapCircleAll(backPos, 0.1f, 64);
11. if (explosiveCollision.Length == 0){
12. GameObject[] balls = new GameObject[splitBalls];
13. float initialAngle = -splitRange / 2;
14. for (int i = 0; i < splitBalls; i++){
15. balls[i] = Instantiate(gameObject, backPos, Quaternion.identity);
16. balls[i].GetComponent<BallMovement>(). setDirection(Quaternion.Euler(0, 0, initialAngle + i \* (splitRange / (splitBalls - 1))) \* bmRef.getDirection());
17. balls[i].GetComponent<BallMovement>(). setBallScore(Mathf.CeilToInt(bmRef.getBallScore() / 2));
18. balls[i].GetComponent<BallMovement>(). setPowerUpID(0);
19. balls[i].GetComponent<SpriteRenderer>().sprite = normalBallSprite;
20. balls[i].transform.localScale = new Vector3(0.7f, 0.7f, 1);
21. for (int j = 0; j < i; j++)
22. Physics2D.IgnoreCollision(balls[i].GetComponent <CircleCollider2D>(), balls[j].GetComponent<CircleCollider2D>());
23. }
24. }
25. Destroy(gameObject);
26. break;
27. }
28. }
29. private void Update(){
30. if (GetComponent<BallMovement>().getBallPowerId() == 3 && particleTimer <= 0.3f)
31. transform.GetChild(0).gameObject.SetActive(true);
32. }
33. private void FixedUpdate(){
34. //biar bom bisa lekat ke target
35. if (bmRef.getBallPowerId() == 3 && collision != null){
36. particleTimer -= Time.deltaTime;
37. transform.position = (Vector2)collision.transform.position - distance;

Segmen Program 7.X (Lanjutan)

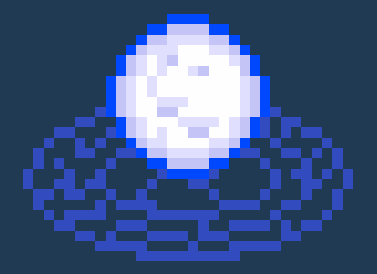
1. }
2. }
3. IEnumerator TimedExplode(float seconds){
4. yield return new WaitForSeconds(seconds);
5. Collider2D[] explosiveCollision = Physics2D.OverlapCircleAll(transform.position, explosionRadius, 8);
6. int hitPlayers = 0;
7. Debug.Log("Boom");
8. foreach (Collider2D coll in explosiveCollision){
9. // Nggak isa melakukan coroutine kalau object ilang
10. coll.GetComponent<SnowBrawler>().getHit(0.5f,gameObject);
11. if (coll.GetComponent<SnowBrawler>().getplayerteam() != bmRef.getPlayerTeam())
12. hitPlayers++;
13. }
14. BarScoreManager.addscore(bmRef.getPlayerTeam(), bmRef.getBallScore() \* hitPlayers);
15. Destroy(gameObject);
16. }
17. }

Dan berikut adalah penjelasannya :

* 2 : Jumlah skor tambahan bila Ice Piercer menembus seseorang
* 3 : Referensi sprite dari bola salju normal, dipakai untuk Stone Auger (bor) saat menembus dinding atau orang
* 4 : Lama bola Explod-o-ball(bom) menempel sebelum meledak
* 5 : Luas Ledakan bola bola Explod-o-ball(bom).
* 6 : Kecepatan dari korban yang kena Freezing Winter (slow), dengan range dari 0 – 1.
* 7 : Lama efek Freezing Winter (slow) bertahan.
* 8 : Jumlah bola yang dibuat dari Stone Auger (bor) yang hancur.
* 9 : Sudut tembakan bola dari Stone Auger (bor) hancur.
* 11 – 15 : Variabel global tambahan sebagai informasi pelengkap.
* 17 – 20 : Setup particletimer dan bmref yang nantinya dipakai untuk timer partikel ledakan dan memudahkan akses ke Script Ball Movement.
* 22 – 77 : Fungsi yang dipakai untuk menjalankan efek bola dan inti dari script ini.
* 24 – 75 : Melihat id dari bola spesial, dan melakukan efek sesuai dengan id power up tersebut.
* 26 – 28 : Skor bola hanya menambah skor sesuai dengan variabel piercescoreadd dan bola dibiarkan.
* 30 – 37 : Bila Snow-a-rang mengenai musuh maka bola tersebut akan mendapat efek seakan ditangkap dan dilempar lagi, sehingga skor dan kecepatan naik sedikit, tetapi dengan pelempar dan kepemilikan tim bola tetap sama. Setelah bola tersebut “tertangkap”, maka bola akan terbang kembali menuju pelempar. Bila Snow-a-rang mengenai tim sama atau dinding maka bola tidak memantul.
* 39 – 45 : Pertama Collider dimatikan agar bola tidak bisa mengenai siapa siapa, kedua ubah body type menjadi static agar melarang Explod-o-ball berinteraksi dengan objek lainnya. Ketiga, tetapkan collision sebagai korban, tentukan jarak antara bola dan korban lalu jalankan Coroutine di baris 91 – 105.
* 47 – 50 : Bila Freezing Winter mengenai seseorang, maka karakter tersebut akan dilambatkan menggunakan fungsi yang terdapat di segmen program 7.X dibawah pada baris XX – XX.
* 52 – 74 : Ini adalah fungsi yang dipakai untuk Stone auger, dimana terdapat beberapa tahap dalam melaksanakan efek bor ini. Diantaranya adalah :
* 53 : Buat sebuah titik yang berjarak 2 unit dari lokasi tubrukan dan masukkan ke variabel backPos.
* 54 : Ambil sudut yang dibuat dari posisi tubruk dan backPos ke dalam variable angle.
* 55 : Buat jarak antara posisi menubruk dan posisi mengeluarkan anak bola menggunakan rumus kuadrat dan sudut variabel angle.
* 56 : Dapatkan posisi dimana anak bola bisa dilepas tanpa menubruk halangan dan masukkan posisi tersebut ke dalam backpos.
* 57 : Nama variabelnya agak ngawur karena copas, tapi kegunaan dari variabel ini adalah untuk mengecek apakah backpos terbaru menubruk sebuah batu di baris 58, dimana bila ada batu yang tertubruk, maka tidak akan terjadi apa apa.
* 59 : Buat array GameObject baru untuk menampung semua bola yang akan dibuat.
* 60 : initialAngle akan menjadi sudut dari bola anak pertama yang ditembak
* 61 – 71 : Membuat anak sejumlah dengan variabel splitBalls.
* 62 : Buat sebuah bola baru di posisi backpos.
* 63- 67 : Tentukan arah terbang, skor bola, powerup ID (0 untuk bola biasa), sprite atau tampilan bola yaitu bola biasa, dan terakhir skala bola yaitu 0.7 x 0.7 kali lebih kecil dari bola biasanya.
* 69 – 70 : Untuk setiap bola yang dibuat, buat bola tersebut mengabaikan bola – bola sebelumnya agar tidak menabrak satu sama lain.
* 73 : Hapus bor bila proses selesai.
* 90 – 104 : Fungsi yang digunakan untuk meledakkan bola setelah beberapa detik.
* 91 : Delay selama beberapa detik sesuai dengan parameter seconds
* 92 : Buat “ledakan” dan simpan setiap karakter yang terkena ledakan tersebut
* 96 – 101 : Untuk setiap karakter yang kena ledakan, berikan efek hitstun terhadap seluruh karakter yang kena, lalu bila karakter yang kena merupakan tim lawan, catat jumlah karakter lawan ke dalam variabel hitPlayers.
* 102 : Tambah skor sesuai dengan jumlah tim lawan yang kena
* 103 : Hancurkan bom

1. Power Up

Class yang terakhir adalah class Power up, yang memiliki perbedaan dengan Class Ball Powerup di Segmen Program 7.X diatas dimana class Ball Power Up terdapat di bola yang dilempar dan berguna untuk mengaktifkan efek khusus bola spesial, sementara Class Power Up terdapat di spawner bola spesial dan bertanggung jawab untuk melakukan spawn bola spesial milik sendiri – sendiri.



Gambar 7.X

Tampilan Spawner Bola Spesial di dalam game.

Berikut potongan kodenya :

Segmen Program 7.X Class PowerUp

1. public class PowerUp : MonoBehaviour{
2. [SerializeField] float spawnTime;
3. [SerializeField] Material playerMaterial;
4. [SerializeField] Color materialColor;
5. [SerializeField] int startingValue;
6. [SerializeField] int ValueRange;
7. [SerializeField] bool randomPowerup;
8. [SerializeField] Sprite[] powerUpSprites;
9. int powerUpValue;
10. float currentSpawnTime;
11. GameObject ball;
12. // Start is called before the first frame update
13. void Start(){
14. if (startingValue > ValueRange || startingValue < 0 || randomPowerup)
15. powerUpValue = Random.Range(1, startingValue + 1);
16. else
17. powerUpValue = startingValue;
18. Material m = new Material(playerMaterial);
19. m.SetColor("\_OutlineColor", materialColor);
20. m.SetFloat("\_OutlineThickness", 1);
21. ball = transform.GetChild(0).gameObject;
22. ball.GetComponent<SpriteRenderer>().sprite = powerUpSprites[powerUpValue-1];
23. currentSpawnTime = 0;
24. }
25. private void Update(){
26. currentSpawnTime -= Time.deltaTime;
27. if (currentSpawnTime <= 0 && !ball.activeSelf){
28. GetComponent<Animator>().speed = 1;
29. ball.SetActive(true);
30. if (randomPowerup){

Segmen Program 7.X (Lanjutan)

1. powerUpValue = Random.Range(1, startingValue + 1);
2. ball.GetComponent<SpriteRenderer>().sprite = powerUpSprites[powerUpValue - 1];
3. }
4. }
5. }
6. public bool isActive(){
7. return currentSpawnTime <= 0;
8. }
9. public (Sprite,int) getPowerupId(){
10. if (currentSpawnTime > 0)
11. return (null,0);
12. ball.SetActive(false);
13. currentSpawnTime = spawnTime;
14. GetComponent<Animator>().speed = 0;
15. return (powerUpSprites[powerUpValue - 1],powerUpValue);
16. }

}

Dan berikut adalah penjelasannya :

* 2 : Lama spawn sebuah bola spesial setelah bola diambil
* 3 : Referensi material untuk memberi outline pada bola salju
* 4 : Warna dari bola spesial (tidak dipakai)
* 5 : Id dari bola spesial pertama, berguna pada tutorial untuk mengatur bola yang keluar pada spawner
* 6 : Sampai berapa range dari id bola spesial yang di-spawn
* 7 : Menentukan apabila spawner akan memanggil bola yang berbeda setiap kali atau bola yang sama.
* 8 : Sprite – sprite dari bola spesial agar bisa mengakses tamppilan bola dengan gampang.
* 17 – 18 : Bila startingValue diluar range valueRange atau randomPowerUp dinyalakan maka di awal dipanggil bola spesial secara acak.
* 18 – 20 : Bila tidak, maka diberikan bola spesial sesuai id
* 21 – 23 : Megatur outline dari bola spesial yang melayang. (Tidak dipakai)
* 24 – 25 : Mengganti sprite dari bola melayang menjadi bola spesial sesuai dengan id bola spesial.
* 26 : Mengatur spawn time menjadi 0 agar bola bisa diambil
* 30 : Timer spawn diturunkan selayaknya countdown
* 31 – 36 : Bila spawner cooldown tetapi timer sudah kembali ke 0, maka diambil bola spesial baru, spawner berada dalam posisi siap diambil dan animasi dijalankan.
* 41 – 43 : Fungsi yang dipakai untuk mengecek apakah bola siap diambil, bila timer <= 0, maka direturn true, bila tidak direturn false.
* 45 – 52 : Fungsi yang dipakai bila ada yang mengambil bola spesial. Bila spawner siap maka bola melayang dihilangkan, animasi dimatikan dan cooldown dimulai lalu direturn id power up dan spritenya.Sementara bila spawner masih dalam cooldown maka tidak dikembalikan apa apa.

1. Karakter Player

Dengan bola dan aspek – aspek yang berhubungan telah dijelaskan di subbab sebelumnya, Sekarang kita akan melihat pelempar dari bola bola tersebut, dimana pada subbab ini akan dijelaskan mengenai apa aksi – aksi yang bisa dilakukan oleh player, dengan subbab selanjutnya menjelaskan mengenai apa yang bisa dilakukan oleh bot.

1. SnowBrawler

Dimulai dengan basis dari karakter kita, atau di bab sebelumnya Dodo, SnowBrawler adalah sebuah Class yang mendasari aksi dari karakter bot dan player, memberi landasan atas apa yang bisa dan tidak bisa mereka lakukan.

Berikut adalah potongan kode yang panjang :

Segmen Program 7.X 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;

Segmen Program 7.X (Lanjutan)

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

Segmen Program 7.X (Lanjutan)

1. caughtBall.GetComponent<BallMovement>(). ballIsCatched(getplayerteam(), ballScoreAdd, ballSpeedAdd, GetComponent<BoxCollider2D>(), gameObject);
2. updateHoldedBallVisuals(false);
3. }
4. else{
5. BallMovement bol = collision.gameObject.GetComponent<BallMovement>();
6. if (bol.getPlayerTeam() != playerteam)
7. BarScoreManager.addscore(bol.getPlayerTeam(), bol.getBallScore());
8. StartCoroutine(getHitNumerator(0.5f, collision.gameObject));
9. bol.trySelfDestruct(gameObject);
10. }
11. }
12. }
13. public void getBall(){
14. int ballindex = SnowBallManager.Instance.getNearestBallIndex(transform);
15. if (ballindex < 0 || Vector2.Distance(transform.position, SnowBallManager.Instance.getBallfromIndex(ballindex).transform.position) > ballTakeRange)
16. return;
17. if (SnowBallManager.Instance.getBallfromIndex(ballindex). GetComponent<PowerUp>()){
18. (ballSprite, ballPowerId) = SnowBallManager.Instance.getBallfromIndex(SnowBallManager.Instance.getNearestBallIndex(transform)).GetComponent<PowerUp>().getPowerupId();
19. if (ballPowerId > 0){
20. displayedBall.GetComponent<SpriteRenderer>().sprite = ballSprite;
21. ballAmount = 1;
22. }
23. }
24. else{
25. int deletedIndex = SnowBallManager.Instance. getNearestBallIndex(transform, ballTakeRange);
26. if (deletedIndex >= 0){
27. ballPowerId = 0;
28. SnowBallManager.Instance.deleteclosestball(transform, ballTakeRange);
29. ballAmount = 1;
30. ballSprite = ball.GetComponent<SpriteRenderer>().sprite;
31. }
32. }
33. SFXSource.clip = AudioScript.audioObject.getSound("Get");
34. SFXSource.Play();
35. updateHoldedBallVisuals(false);
36. }
37. public void shootBall(Vector2 direction){

Segmen Program 7.X (Lanjutan)

1. GameObject ballin;
2. if (caughtBall != null){
3. ballin = caughtBall;
4. ballin.GetComponent<BallMovement>(). setDirection(direction);
5. ballin.transform.position = this.transform.position;
6. ballin.SetActive(true);
7. caughtBall = null;
8. }
9. else{
10. ballin = Instantiate(ball, (Vector2)this.transform.position, Quaternion.identity);
11. ballin.GetComponent<BallMovement>(). initialize(throwSpeed, direction, playerteam, ballScoreInitial, this.GetComponent<BoxCollider2D>(), gameObject, ballPowerId);
12. ballAmount--;
13. if (ballPowerId > 0)
14. ballin.GetComponent<SpriteRenderer>().sprite = ballSprite;
15. }
16. SFXSource.clip = AudioScript.audioObject.getSound("Yeet");
17. SFXSource.Play();
18. ballin.GetComponent<SpriteRenderer>().material = GetComponent<SpriteRenderer>().material;
19. updateHoldedBallVisuals(true);
20. }
21. public int getBallAmount(){
22. return ballAmount + ((caughtBall == null) ? 0 : 1);
23. }
24. public void slowDown(float movementSpeedSlow, float slowTime){
25. StartCoroutine(slowDownNumerator(movementSpeedSlow, slowTime));
26. }
27. IEnumerator slowDownNumerator(float slowPower, float seconds){
28. GetComponent<SpriteRenderer>().color = new Color(11 / 255, 211 / 255, 1);
29. runSpeed = originalRunSpeed \* slowPower;
30. yield return new WaitForSeconds(seconds);
31. GetComponent<SpriteRenderer>().color = Color.white;
32. runSpeed = originalRunSpeed;
33. }
34. public void getHit(float seconds, GameObject snowBall){
35. StartCoroutine(getHitNumerator(seconds,snowBall));
36. }
37. public IEnumerator getHitNumerator(float seconds,GameObject snowBall){
38. if (snowBall.GetComponent<BallMovement>().getPlayerTeam() != playerteam){
39. GameObject numbers = Instantiate(numberReference);
40. numbers.GetComponent<NumbersController>(). setGambar(snowBall.GetComponent<BallMovement>().getBallScore());
41. numbers.GetComponent<NumbersController>(). StartingPosition = transform.position;
42. }
43. canAct = false;
44. animator.SetBool("IsHit", true);
45. Debug.Log("Kena Hit");
46. yield return new WaitForSeconds(seconds);
47. canAct = true;
48. animator.SetBool("IsHit", false);
49. Debug.Log("Selesai Kena Hit");
50. }
51. public IEnumerator catchBall(){
52. runSpeed = 0;
53. iscatching = true;
54. canCatchBall = false;
55. animator.SetBool("IsCatching", true);
56. yield return new WaitForSeconds(ballCatchTimer);
57. animator.SetBool("IsCatching", false);
58. runSpeed = originalRunSpeed;
59. iscatching = false;
60. StartCoroutine(catchRecharging());
61. }
62. public IEnumerator catchRecharging(){
63. GetComponent<SpriteRenderer>().color = new Color(0.5f, 0.5f, 0.5f);
64. yield return new WaitForSeconds(catchRecharge);
65. GetComponent<SpriteRenderer>().color = new Color(1, 1, 1);
66. canCatchBall = true;
67. }
68. public IEnumerator shartShooting(){
69. animator.SetBool("isShooting", true);
70. canAct = false;
71. runSpeed = 0;
72. yield return new WaitForSeconds((0.5f \* 5) / 6);
73. runSpeed = originalRunSpeed;
74. canAct = true;
75. updateHoldedBallVisuals(true);
76. animator.SetBool("isShooting", false);
77. }
78. public void tryCatch(){
79. if (!iscatching)
80. StartCoroutine(catchBall());
81. }
82. void updateHoldedBallVisuals(bool isThrown){
83. if (GetComponent<DisplayBall>() != null)
84. GetComponent<DisplayBall>().updateUI(isThrown);
85. if (caughtBall == null && ballAmount == 0){

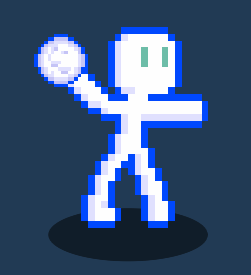
Segmen Program 7.X (Lanjutan)

1. displayedBall.SetActive(false);
2. return;
3. }
4. GetComponent<Animator>().enabled = false;
5. displayedBall.SetActive(true);
6. GetComponent<Animator>().enabled = true;
7. if (caughtBall != null){
8. displayedBall.GetComponent<SpriteRenderer>().sprite = caughtBall.GetComponent<SpriteRenderer>().sprite;
9. return;
10. }
11. displayedBall.GetComponent<SpriteRenderer>().sprite = ballSprite;
12. }
13. }

Dan berikut adalah penjelasannya, agar tidak memperpanjang buku maka beberapa potongan kode akan dipersingkat penjelasannya;

* 3 – 31 : Variabel – variabel yang mengatur cara kerja dari karakter kita, dari kecepatan jalan, jumlah bola yang dipegang, tim dari karakter tersebut dst.
* 33 – 41 : Pengaturan awal beberapa variabel
* 44 – 46 : Secara periodik fungsi yang di dalam if akan dijalankan.
* 47 – 49 : Mengatur variabel yang dipakai untuk animasi gerakan.
* 50 – 51 : Mempersiapkan lastpos untuk pengaturan animasi selanjutnya, lalu mereset timer untuk memanggil fungsi ini.
* 53 : Mengatur variabel isAiming tergantung dari status membidik
* 57 – 73 : Bila karakter berturbrukan dengan sebuah bola, maka fungsi didalam akan dijalankan
* 58 – 65 : Bila karakter mencoba menangkap bola, maka dilihat apakah karakter punya bola yang ditangkap, bila ada bola tersebut dibuang, lalu bola yang barusan ditangkap, disimpan dan diatur variabelnya.
* 66 – 72 : Tapi bila karakter tidak sedang menangkap, maka dilihat apabila dia setim atau tidak dengan pelempar bola, bila setim, maka bola hanya melakukan hitstun, tapi bila tidak setim, maka akan diberi skor juga setelah memberi hitstun. Setelah ini bola akan mencoba menghancurkan diri menggunakan fungsi trySelfDestruct di segmen program 7.X.
* 76 – 99 : Fungsi yang dipakai untuk mengambil gumpalan bola atau bola spesial.
* 77 : Diambil index bola menggunkan fungsi milik SnowBallManager (segmen 7.X)
* 78 – 79 : Bila tidak ditemukan bola, maka fungsi direturn awal.
* 80 – 86 : Bila bola yang ditemukan adalah sebuah bola spesial, maka diambil powerId dan gambar dari bola tersebut, lalu tambahkan bola sebanyak 1.
* 87 – 95 : Sementara bila bola adalah bola biasa, maka dicek apakah bola terhapus. Bila terhapus maka jumlah bola akan ditambah dengan powerId 0.
* 96 – 97 : Mainkan suara mengambil bola
* 98 : Ubah bola di tangan sesuai dengan bola yang didapat.
* 101 – 121 : Fungsi yang dipakai untuk menembak bola.
* 103 – 109 : Bila karakter memiliki bola yang dari tangkapan, maka bola tersebut akan dilempar duluan dan bola tertangkap hilang.
* 110 – 116 : Tetapi bila karakter hanya memiliki bola yang diambil, maka bola tersebut yang dilempar, jadi prioritas lemparan adalah bola yang ditangkap.
* 117 – 118 : Mainkan suara melempar bola.
* 119 : Beri bola yang dilempar outline yang sama dengan karakter pelempar
* 120 : Update tampilan bola yang dipegang setelah melempar bola
* 123 – 125 : Mereturn jumlah bola yang dipegang karakter.
* 127 – 129 : Melakukan Coroutine di baris 131 – 137 yang memperlambat gerakan karakter.
* 132 – 134 : Ganti warna karakter yang terkena efek lambat, lambatkan gerakan karakter tersebut lalu biarkan efek ini berjalan sesuai dengan parameter seconds.
* 135 – 136 : Kembalikan warna dan kecepatan karakter menjadi normal.
* 139 – 156 : Sebuah fungsi yang dipakai bila karakter terkena sebuah bola
* 144 – 148 : Bila karakter yang kena bukan tim pelempar, maka munculkan angka yang memberitahu jumlah skor yang didapat.
* 149 – 152 : Karakter yang terkena lemparan tidak bisa beraksi untuk sesaat
* 164 – 167 : Buat karakter bisa bergerak lagi.
* 158 – 168 : Coroutine yang dijalankan apabila karakter mencoba menangkap bola, cara kerja mirip seperti Coroutine getHitNumerator diatas.
* 170 – 175 : Berikan karakter warna gelap apabila tidak bisa menangkap.
* 177 – 186 : Fungsi yang dipakai untuk melempar bola, mirip dengan Coroutine getHitNumerator dan catchBall.
* 188 – 191 : Bila karakter sedang tidak menangkap, maka dijalankan Coroutine diatas
* 193 – 208 : Fungsi yang digunakan untuk mengganti tampilan dari bola yang dipegang oleh karakter dalam game.
* 194 – 195 : Bila karakter dikontrol oleh player, maka update tampilan bola dalam UI.
* 196 – 199 : Bila karakter tidak ada bola maka tidak ada bola yang ditampilkan
* 200 – 202 : Tampilkan bola di tangan karakter
* 203 – 206 : Bila bola hasil tangkapan, maka ambil sprite dari bola tangkapan tersebut dan pasang di bola di tangan.
* 207 : Bila tidak ada bola hasil tangkapan ambil sprite yang disimpan dalam Class ini.

Untuk visualisasi, ini adalah tampilan dari karakter yang memegang bola :



Gambar 7.X

Tampilan Karakter membidik sebuah bola

1. Movement / Gerakan

Segmen Program 7.X Class Player Movement

1. public class PlayerMovement : MonoBehaviour{
3. Vector2 moveDirection;
4. Rigidbody2D thisRigid;
5. ShootMechanic SMReference;
6. SnowBrawler SBReference;
7. // Start is called before the first frame update
8. void Start(){
9. moveDirection = new Vector2(0,0);
10. thisRigid = this.GetComponent<Rigidbody2D>();
11. SMReference = this.GetComponent<ShootMechanic>();
12. SBReference = this.GetComponent<SnowBrawler>();
13. thisRigid.useFullKinematicContacts = true;
14. }
15. // Update is called once per frame
16. void Update(){
17. if (!SBReference.canAct){
18. moveDirection = new Vector2(0, 0);
19. return;
20. }
21. float diagonalCheck = Mathf.Sqrt(Mathf.Pow(Input.GetAxisRaw("Horizontal"), 2) + Mathf.Pow(Input.GetAxisRaw("Vertical"), 2));
22. moveDirection.x = Input.GetAxisRaw("Horizontal") \* SBReference.runSpeed ;
23. moveDirection.y = Input.GetAxisRaw("Vertical") \* SBReference.runSpeed;
24. if (diagonalCheck != 0f)
25. moveDirection /= diagonalCheck;
26. if (SMReference.isAiming)
27. moveDirection \*= SMReference.aimMovementSpeedPerc;
28. if (Input.GetAxisRaw("Horizontal") != 0 && !PauseGame.isPaused && !SMReference.isAiming)
29. transform.localScale = new Vector3(Input.GetAxisRaw("Horizontal"), 1, 1);
30. }
31. private void FixedUpdate(){
32. thisRigid.MovePosition((Vector2)this.transform.position + moveDirection \* Time.deltaTime);
33. }
34. }
35. Shoot Mechanic

Segmen Program 7.X 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. }
88. Catch Ball

Segmen Program 7.X Class CatchBall

1. public class CatchBall : MonoBehaviour{
2. SnowBrawler brawlerReference;
3. private void Start(){
4. brawlerReference = GetComponent<SnowBrawler>();
5. }
6. // Update is called once per frame
7. void Update(){
8. if (Input.GetKeyDown(KeyCode.Mouse1) && brawlerReference.canAct && !brawlerReference.isAiming && brawlerReference.canCatchBall)
9. brawlerReference.tryCatch();
10. }
11. }
12. Karakter Musuh
13. Bot Actions
14. Coordinate Movement
15. State Machine
16. Sistem Timer