Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202072633
Vol. 7, No. 2, April 2020, hlm. 385-390 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

PENENTUAN JARAK KARAKTER PEMAIN DENGAN KARAKTER-BUKAN-PEMAIN YANG HARUS MENGIKUTI PEMAIN PADA PENGEMBANGAN GAME SIMULASI MITIGASI BENCANA ERUPSI GUNUNG

Henny Widowati[1], Rachma Putri Widhowati[2], Sulistyo Puspitodjati[3], D. L. Crispina Pardede*[4]

[1]Departemen Program Magister Manajemen Sistem Informasi, Universitas Gunadarma
[2]Departemen Sistem Informasi, Universitas Gunadarma
[3,4] Departemen Teknik Informatika, Universitas Gunadarma
Email: [1]widowati@staff.gunadarma.ac.id, [2]rachmaputri.widhowati@gmail.com,
[3]sulistyo@staff.gunadarma.ac.id, [4]pardede@staff.gunadarma.ac.id
*Penulis Korespondensi

(Naskah masuk: 30 Oktober 2018, diterima untuk diterbitkan: 11 Februari 2020)

Abstrak

Game simulasi sering melibatkan karakter pemain (*player character*) dan karakter bukan pemain (*non player character*) (KBP/NPC), sebagaimana pada *game* simulasi mitigasi bencana erupsi gunung yang sedang dikembangkan. Salah satu bentuk interaksi antara karakter pemain dengan KBP adalah KBP harus mengikuti karakter pemain. Pemrograman agar KBP mengikuti pemain, perlu menentukan jarak sejauh apa sehingga KBP secara otomatis mengikuti pemain. Penentuan jarak antara karakter pemain dengan KBP menjadi bahasan dalam makalah ini. Jarak dihitung secara matematis menggunakan konsep pitagoras, dan dengan pertimbangan estetika visual. Salah satu unsur visual yang dipertimbangkan adalah bentuk, baik KBP maupun bentuk karakter pemain. Jarak yang merupakan hasil perhitungan digunakan sebagai batasan untuk menganimasi (menggerakkan) KBP mengikuti karakter pemain. Animasi KBP, yaitu karakter berupa sapi dan manusia mengikuti pemain adalah bagian dari pengembangan *game* simulasi mitigasi bencana erupsi gunung berapi. Pengujian dengan membuat karakter pemain dan *collider* yang ada pada KBP bertumbukan atau bersentuhan berhasil membuat KBP bisa mengikuti karakter pemain sesuai dengan jarak yang ditentukan, dan berhasil mencapai tujuan.

Kata kunci: *game, karakter bukan pemain, estetika visual*.

DISTANCES DETERMINATION OF PLAYERS TO NON-PLAYERS CHARACTERS WHO MUST FOLLOW PLAYERS IN THE DEVELOPMENT OF VOLCANIC ERUPTION MITIGATION SIMULATION GAMES

Abstract

Simulation games often involve player characters and non-player characters (NPC), as in the mountain eruption disaster mitigation simulation game developed. One form of interaction between player characters and NPC is that NPC must follow the player character. Programming so that NPC follows players requires determining the distance so that NPC automatically follows the player. Determination of the distance between player characters (players) and non-player character objects (NPC) is discussed in this paper. The distance is measured mathematically using the pythagorean concept and visual aesthetics consideration. One visual element considered is the shape, both NPC and the player's own shape. The resulting distance is used as boundary to animate (move) NPC following the player's character. NPC's animation: the character of cows and humans following players is part of the development of mitigation simulation games for volcanic eruptions. The test, by making the player character and the collider on the NPC collide, succeeded in making the NPC follows the player character according to the specified distance, and succeeded in achieving the goal.

Keywords: *game, non player character, visual aesthetic.*

1. PENDAHULUAN

Game Simulasi umumnya adalah *game* yang dimaksudkan menyimulasikan kehidupan nyata (Salmi, 2015). *Game* Simulasi Mitigasi Bencana Erupsi Gunung Berapi merupakan *game* simulasi

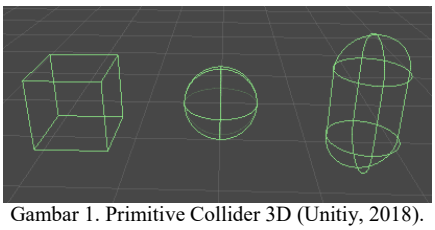
yang sedang dikembangkan, dan dimaksudkan sebagai salah satu alat pembelajaran mitigasi bencana gunung berapi saat gunung berapi meletus. *Game* dikembangkan mengingat Indonesia terletak di atas cincin api, dan bencana yang telah terjadi banyak menelan korban (Indonesia Invesments News Letters, 2018). *Game* yang sedang dikembangkan tersebut berlatar kejadian di sekitar desa di kaki gunung berapi yang ada di Indonesia ketika bencana gunung api meletus. Karena dimaksudkan mengikuti kejadian nyata di desa, maka salah satu misi dalam permainan adalah karakter pemain (*player*), yang digerakkan pemain, menyelamatkan karakter bukan pemain (KBP) atau NPC (*non player characters*). KBP dalam permainan yang dikembangkan dan yang dibahas dalam makalah ini, adalah warga dan hewan ternak. Pemain akan menang dalam misi tersebut, jika menyelamatkan warga dan hewan ternak sapi dan selamat dari rintangan yang muncul akibat gunung berapi meletus. Pola animasi sapi (KBP) mengikuti *player* telah dikembangkan dalam *game* (Pardede, *et al.*, 2017), namun penentuan jarak terhadap KBP sapi dan KBP lain secara umum belum ditentukan. Makalah ini membahas salah satu unsur dalam pengembangan *game* tersebut, yaitu penentuan jarak antara karakter pemain dengan KBP, sehingga KBP mengikuti pemain. Penentuan jarak didasarkan pada rumus matematik dengan pertimbangan estetika visual. Jika jarak terlalu jauh, objek yang satu tidak tampak mengikuti objek yang seharusnya diikuti, namun jika jarak terlalu dekat akan tampak tidak nyata secara visual. Estetika dalam permainan mencakup aspek permainan yang dialami oleh pemain secara langsung melalui audio dan grafik atau secara tidak langsung melalui aturan, geografi, karakteristik temporal dan jumlah pemain. (Gupta & Baumie, 2014)

Pengembangan *game* tiga dimensi (3D) melibatkan banyak objek 3D. Objek 3D tersebut akan bergerak pada suatu peta (*map*) lingkungan *game*. Salah satu komponen objek pada 3D program adalah posisi. Posisi diidentifikasi melalui sumbu XYZ. Sumbu X untuk menyatakan gerak objek dari kiri ke kanan (horisontal), gerak objek ke atas dan ke bawah dinyatakan dengan sumbu Y, sumbu Z untuk merepresentasikan gerak menjauh dan mendekat objek.

Jika objek-objek tersebut saling berinteraksi (bertemu secara pasif atau aktif), maka diperlukan pengenal bahwa mereka sudah saling "bertemu". Pengenal tersebut disampirkan atau diselimutkan pada badan objek. Selimut tersebut pada perangkat lunak pemodel 3D dikenal dengan istilah *collider*. Sebuah *collider* pada objek nantinya tidak tampak, dan dapat menggunakan bentuk yang tidak sama dengan objek, namun tetap mempertimbangkan efisiensi animasi.

Collider pada Unity 3D yang paling sederhana adalah yang biasa disebut sebagai tipe *collider primitive*, karena mengikuti bentuk primitif kubus (*Box Collider*), bola (*Sphere Collider*), dan tabung kapsul (*Capsule Collider*) (Gambar 1 adalah *primitive collider* pada Unity 3D (Unitiy, 2018)). Terdapat *collider* khusus pada objek 2D, yaitu *Box Collider* 2D dan *Circle Collider* 2D. Beberapa *collider* dapat digunakan pada sebuah objek untuk membuat *collider* gabungan.

Objek 3D pada Unity dapat mempunyai komponen *Rigidbody*. *Collider* dapat ditambahkan pada sebuah objek tanpa atau dengan komponen *Rigidbody*. Objek tanpa *Rigidbody* biasanya digunakan pada objek lantai, tembok, dan elemen-elemen statis lainnya pada *scene*. *Collider* tersebut disebut *collider* statis, dan biasanya dilampirkan pada objek yang tidak memerlukan komponen *Rigidbody*. *Collider* pada sebuah objek yang memiliki *Rigidbody* diistilahkan sebagai *collider* dinamis. *Collider* statis dapat berinteraksi dengan *collider* dinamis tapi karena mereka tidak memiliki *Rigidbody*, mereka tidak akan bergerak dalam merespon pertemuan (tabrakan). *Collider* pada karakter pemain, sapi dan warga adalah *collider* dinamis, sementara lantai, dinding, sofa adalah *collider* statis.



Gambar 1. Primitive Collider 3D (Unitiy, 2018).

Collision detection merupakan fungsi pada Unity untuk mendeteksi suatu pertemuan (tabrakan) antara 2 atau lebih objek. Pertemuan antar *collider* untuk menentukan apa yang akan terjadi pada objek yang menabrak atau pada objek yang ditabrak ataupun yang menabrak (Akenine-Möller *et al*., 2018). *Collision detection* ini juga berguna untuk menentukan posisi dari satu objek dengan objek yang lain agar tidak ada objek yang saling menembus, sehingga *game* yang dibuat memiliki kesamaan dengan realita yang ada. Dua macam teknik pendeteksian tumbukan, adalah *priori detection* dan *post detection*.

Jika *collider* digunakan sebagai pemicu, misalkan objek sudah mencapai area tertentu dan perlu memunculkan *cutscene*, maka dapat menggunakan fungsi "*Is T*rigger." Jika yang diinginkan adalah objek yang bereaksi terhadap tabrakan fisik dengan objek lain, maka perlu menambahkan komponen *Rigidbody*. Untuk tabrakan bukan pemicu digunakan *OnCollisionEnter*, *OnCollisionExit*, dan *OnCollision Stay*.

OnCollisionEnter dipanggil saat *collider* objek menyentuh *collider* objek lain yang mempunyai *Rigidbody*. Saat sedang bersentuhan (bertabrakan), *OnCollisionStay* dipanggil. Jika sentuhan diakhiri, On*Col*lisionExit dipanggil.

Cara yang sama berlaku jika objek sebagai pemicu (diatur sebagai Tr*igg*er), dan yang digunakan adalah OnT*rigg*erEnter, OnT*rigger*Stay, dan *OnTriggerExit*.

2. METODE PENELITIAN

Game Simulai Mitigasi Bencana Erupsi Gunung, terdiri dari beberapa misi. Misi yang memerlukan animasi karakter bukan pemain mengikuti pemain, adalah misi dimana pemain harus menyelamatkan hewan ternak sapi, dan "mengajak" sapi untuk mengikutinya sampai posisi yang ditentukan. Begitu juga dalam misi penyelamatan keluarga dan warga, pemain harus "mengajak" mereka ke titik tertentu, untuk dinilai sebagai misi yang berhasil. Deteksi *collider* yang digunakan adalah objek yang diatur sebagai pemicu (*Trigger*), karena kedua objek (pemain dan sapi atau warga), tanpa harus bersentuhan, mereka sudah harus bereaksi, yaitu mengikuti pemain.

Objek karakter bukan pemain (KBP) berjalan mengikuti karakter pemain jika memenuhi jarak tertentu. Objek pertama yang diperhatikan adalah objek karakter bukan pemain. Untuk estetika visual kewajaran seekor sapi mengikuti orang (karakter pemain dalam *game*) adalah sedekat-dekatnya sekitar 40 cm dan sejauh-jauhnya adalah 2 m. Hal ini juga mempertimbangkan besar sapi relatif terhadap manusia. Diasumsikan, objek sapi pada *game* adalah sapi besar dengan panjang badan 1,5 m sampai 2 m. Sehingga jarak sekitar 2 m masih dianggap wajar, dan jarak terdekat agar dapat dinilai dari segala arah tidak berkesan "tabrakan".

Jika panjang satu grid peta lingkungan *game* mewakili ukuran *x* cm pada dunia nyata (skala 1: *x*) dan jika posisi sapi dihitung pada pusat badan sapi yang merupakan titik tengah panjang badan (*PjBdn*) (sekitar perut sapi), dan panjang kepala sapi (*LrKpla*) sekitar 20 cm, maka panjang dari setengah badan sapi sampai ke kepala adalah (1/2 * 200 cm) + 20 cm = 120 cm. Jika ditambah jarak terdekat dengan pemain (*JrkRielMin*) yang adalah 40 cm, maka jarak minimum (*JrkMin*) dengan pemain adalah 160 cm. Jika ukuran grid peta yang digunakan adalah 20 cm, maka jarak terdekat objek karakter pemain dengan objek sapi adalah 160 cm/20 cm = 8 satuan. Secara umum, persamaan (1) merupakanpenentuan jarak minimum antara objek pemain dengan objek KBP (dalam contoh di atas adalah sapi).

$$JrkMin = \frac{\{(.5 * PjBdn + LrKpla) + (JrkRielMin)\}}{x} \quad (1)$$

Jarak terjauh (*JrkMax*) agar objek KBP mengikuti objek pemain ditentukan berdasarkan persamaan (2).

$$JrkMax = \frac{\{(.5 * PjBdn + LrKpla) + (JrkRielMax)\}}{x} \quad (2)$$

Jika sapi dianggap sapi jinak yang sangat menurut sehingga dari jarak yang mendekati 4 m (*JrkRielMax*), (380 cm dari muka sapi) sudah dapat mengikuti pemain, maka dengan asumsi seperti di atas ((.5 * 2 m) + 20 cm) + 380 cm = 500 cm adalah jarak terjauh dari tengah badan sapi atau ekivalen dengan 500 cm/20cm = 25 satuan pada program model 3D.

Jarak antar dua vektor a = (a_x, a_y, a_z) dan b = (b_x, b_y, b_z) berdimensi tiga dihitung menggunakan konsep norm selisih vektor a-b (Larson, 2017). Persamaan (3) merupakan konsep Eulerian, salah satu konsep yang umum digunakan.

$$\|\bar{a} - \bar{b}\| = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2} \quad (3)$$

Kecepatan objek KBP dapat diatur sesuai dengan sifat dari objek hewan itu sendiri dan objek pemain. Sapi adalah hewan yang dapat berjalan mengikuti kecepatan rata-rata berjalan manusia. Kecepatan rata-rata umumnya manusia adalah 100 langkah per menit. Jika satu langkah manusia diasumsikan sama dengan 30 cm, maka kecepatan rata-rata langkah manusia adalah 3000 cm per 60 detik atau 50 cm perdetik. Dalam kasus *game* ini, objek sapi mengikuti pemain objek pemain diatur pada nilai 2,5 satuan per detik (karena satu grid peta adalah 20 cm).

Pembuatan *script* jarak antara karakter pemain dan KBP (*Non Player Character*) dilakukan pada Unity menggunakan bahasa C#. *Script* perhitungan digunakan untuk menghitung jarak antara karakter pemain dengan KBP yaitu penduduk dan hewan ternak. Perhitungan jarak tersebut nantinya akan dipergunakan sebagai kondisi *mission complete* pada *game*, apakah misi yang sedang dimainkan berhasil atau tidak. Untuk mendukung berjalannya perhitungan jarak antara karakter pemain dan KBP, maka disusun *script*.

Berikut adalah *script* penentuan jarak Ge*tD*istance() yang digunakan untuk menggerakkan objek yang mengikuti objek karakter pemain:

```
private void GetDistance(){
thisPos = transform.position;
playerPos = player.position;
currentDistance= Vector3.Distance (thisPos, playerPos);
}
```

GetDistance merupakan bagian dari *AIFollowPlayer*.cs. *AIFollowPlayer*.cs mengandung beberapa fungsi lain selain *GetDistance*, yaitu: *Awake*(), untuk menginisialisasi pada awal permainan *isMoving = false*. Terdapat pula fungsi Update(), untuk memeriksa kapan KBP (penduduk dan sapi) akan bertabrakan (*collision detection*) dengan karakter pemain dan apakah sudah mencapai tujuan atau belum. Untuk memeriksa jarak minimal dan maksimal dengan karakter, dan untuk mengatur animasi *idle* (diam) dan jalannya KBP. Fungsi lain

adalah *Move*() untuk memberi perintah *GameObject* KBP bergerak atau berjalan.

AIFollowPlayer.cs juga melibatkan fungsi *LookAtPlayer*() untuk merotasi KBP agar melihat ke arah dimana karakter berada.

Pada *Script AIFollowPlayer*.cs terdapat beberapa variabel yang masing-masing mempunyai fungsi yang berbeda. Variabel tersebut adalah *current distance* untuk melihat jarak antara karakter dengan KBP saat ini. Variabel *speed* berguna untuk menentukan kecepatan gerak dari KBP. *Min jarak* dan *max jarak* adalah dua variabel yang berguna untuk menentukan nilai minimum dan maksimum jarak yang harus dipenuhi agar KBP dapat tetap mengikuti karakter pemain untuk menuju ke tujuan yang sudah ditentukan sehingga bisa menyelesaikan misi. Keempat variabel di atas memiliki tipe data *float*, yang berarti nilai dari variabel dalam bilangan pecahan.

Selain itu, variabel *is collide* berfungsi untuk menandai apakah karakter pemain sudah bertabrakan dengan *collider* dari KBP atau belum untuk menerima misi. *Is reach goal* berfungsi untuk menandai apakah karakter pemain berhasil menyelesaikan misi. Selanjutnya, *is moving* berfungsi untuk menandai apakah KBP sudah bisa berjalan atau belum. Berikut adalah bagian dari *script* untuk menggerakkan KBP mengikuti karakter pemain.

```
private void Update(){
        if(isCollide && !isReachGoal &&
          NPCMission.Instance.isPicked)
        LookAtPlayer ();
        if (isCollide && currentDistance < maxJarak &&
          currentDistance >= minJarak &&
          NPCMission.Instance.isPicked) {
          if (!isReachGoal) {
              Move ();
              }
          } else {
              isMoving = false;
              }
          anim.SetBool ("Jalan", isMoving);
          GetDistance ()
          private void Move(){
              isMoving = true;
              CharacterController controller =
                  GetComponent<CharacterController>();
              Vector3 forward = transform.TransformDirection
                  (Vector3.forward);
              controller.SimpleMove (forward * speed);
              if(!controller.isGrounded) //gravity effect
                  controller.Move (new Vector3 (0f, -9.8f, 0f));
          }
private void LookAtPlayer(){
          Vector3 targetPos = new Vector3 (player.position.x,
              transform.position.y, player.position.z);
  //      transform.LookAt (targetPos);
          var neededRotation = Quaternion.LookRotation(targetPos
              - transform.position);
              transform.rotation = Quaternion.Slerp
              (transform.rotation, neededRotation, Time.deltaTime);
              }
private void GetDistance(){
      thisPos = transform.position;
      playerPos = player.position;
      currentDistance = Vector3.Distance (thisPos, playerPos);
          }
```

```
private void OnTriggerEnter(Collider col){
        if (col.gameObject.CompareTag ("Player")) {
            isCollide = true;
        } else if (col.gameObject.CompareTag ("Tujuan")) {
            isReachGoal = true;
            isMoving = false;
            GoalMisi.Instance.AddCount (1);
        }
    }
}
```

3. HASIL DAN PEMBAHASAN

Hasil penggunaan persamaan (3) dan penggunaan batasan sesuai persamaan (1) dan (2) yang diimplementasikan pada *Unity* menggunakan C# dapat dilihat pada Gambar 2. Gambar tersebut menunjukkan penampakan visual ketika objek sapi belum mengikuti karakter pemain, karena objek karakter pemain belum mendekati KBP sapi sebagaimana jarak yang dihitung menggunakan persamaan (1) dan (2).

Gambar 2. Sapi Belum Mengikuti Karakter

Ketika jarak minimum 8 satuan unit dipenuhi, maka sapi mengikuti karakter pemain sebagaimana terlihat pada Gambar 3, dan masih megikuti pemain sampai pengaturan jarak terjauh 25 unit sebagaimana pada Gambar 4.

Gambar 3. Sapi Mengikuti Karakter Pemain Pada Jarak Minimum (8 Unit)

Ketika karakter pemain bergerak terlalu jauh, maka sapi berhenti sebagaimana terlihat pada Gambar 5.

Hal yang sama terjadi ketika persamaan (1) dan (2) dengan lebar kepala LrKpla = 0 diterapkan pada KBP penduduk (objek manusia). Baik saat jarak masih terlalu jauh (Gambar 6), saat jarak terdekat dan saat jarak terjauh dapat dilihat masing-masing, pada Gambar 7 dan Gambar 8.

Gambar 4. Sapi Mengikuti Karakter Pemain Pada Jarak Maksimum (25 Unit)

Gambar 5. Sapi Berhenti Mengikuti Karakter (Karena Jarak Lebih Dari 25 Unit)

Gambar 6. Penduduk Belum Mengikuti Karakter Pemain.

Gambar 7. Penduduk Mengikuti Karakter Pemain Pada Jarak Minimum 6 Unit.

Gambar 8. Penduduk Mengikuti Karakter Pemain Pada Jarak Maksimum 25 Unit.

Begitu juga contoh penampakan visual dapat dilihat pada Gambar 9, ketika karakter pemain menjauh di luar jarak pada pengaturan persamaan (1) dan (2).

Uji coba dilakukan dari segi grafik tampilan, dan kesesuaian objek ketika dijalankan.

Misi penyelamatan dapat dimainkan dengan baik hingga selesai. Proses uji coba dengan membuat karakter dan *collider* yang ada pada KBP bertumbukan atau bersentuhan berhasil membuat KBP bisa mengikuti karakter menuju tujuan sesuai dengan jarak yang telah ditentukan, yaitu jarak minimum 8 unit dan maksimum 25 unit. KBP juga sudah bisa mengikuti karakter pemain hingga ke tujuan yang telah ditentukan.

Gambar 9. Penduduk Berhenti Mengikuti Karakter Pemain Karena Jarak Lebih Dari Jarak Maksimum.

Game dikembangkan untuk dapat dijalankan pada perangkat dengan resolusi yang minimum sekalipun, karena itu *collider* yang diterapkan pada karakter pemain maupun karakter bukan pemain adalah *collider* primitif tunggal. Animasi karakter pemain mendekati sapi atau penduduk sampai dapat menyentuh badan (misalnya mengelus sapi atau menggandeng penduduk) belum diterapkan pada pengembangan *game* ini.

Pengaturan jarak maksimum dan minimum pada pengembangan animasi KBP mengikuti pemain ini masih diinputkan secara manual belum dimasukkan sebagai bagian dari atribut karakter pemain maupun karakter bukan pemain.

4. KESIMPULAN

Secara visual permainan, animasi karakter pemain mengikuti karakter bukan pemain (KBP) sebagai bagian dari *game* Simulasi Mitigasi Bencana Erupsi Gunung Berapi, sudah dapat di-implementasikan dengan baik menggunakan Unity dengan script C#. Saat kapan KBP mengikuti pemain atau tidak, ditentukan menggunakan jarak minimum dan maksimum yang dihitung menggunakan rumus jarak Euclidean dan dengan menimbang estetika visual melalui bentuk dan besar KBP relatif terhadap karakter pemain.

Pengembangan selanjutnya adalah menambah atribut KBP berupa ukuran badan dari KBP, seperti: lebar badan dan lebar kepala. Sehingga, rumus dapat diberlakukan untuk semua objek KBP yang dilibatkan dalam *game* kami maupun game lain secara umum. Pengembangan pada animasi KBP mengikuti karakter pemain juga dapat dilakukan dengan menambah sistem *collider* gabungan, namun masih dengan *time complexity* maupun *space complexity* yang efisien dan efektif, sehingga dapat dijalankan pada perangkat dengan spesifikasi minimum.

DAFTAR PUSTAKA

AKENINE-MÖLLER, T., HAINES, E., HOFFMAN, N., PESCE, A., HILLAIRE S., & IWANICKI, M., 2018. Real-Time Rendering, 4th Edition. A K Peters/CRC Press.

GUPTA, D. & KIM, B., 2014. Aesthetic Design For Learning With Games. Proceedings of the 22nd International Conference on Computers in Education, ICCE 2014.

Indonesia Investment News Letter, 2018. Natural Disasters in Indonesia. Retrieved from https://www.indonesia-investments.com/ business/risks/natural-disasters/item243?. [Diakses 2 November 2018]

LARSON, R., 2017. Elementary Linear Algebra, 8th Edition. Cengage Learning. USA

PARDEDE, D.L.C., MARLIE, E., PUSPITODJATI, S., & WIDOWATI, H., 2017. Animating Cows to Follow Player. Journal of Engineering and Applied Sciences, 12 (8), pp. 2189-2193.

SALMI, J., 2015. Learning Simulation Game Development Case: Simulation Game for EPCM Project Management Training, Aalto University, Espoo, Finlandia.

UNITY, 2018. Colliders. https://docs.unity3d.com/ Manual/CollidersOverview.html, [Diakses 9 Juli 2018].