Journal of Engineering and Applied Sciences 12 (8): 2189-2193, 2017
ISSN: 1816-949X

Animating Cows to Follow Player

[1]D.L. Crispina Pardede, [2]Escher Marlie, [3]Sulistyo Puspitodjati and [2]Henny Widowati
[1]Department of Computer System,
[2]Department of Information System,
[3]Department of Informatics Engineering, Gunadarma University, Depok, Indonesia

Abstract: We are creating volcanic eruption disaster mitigation simulation game. The game supposedly to give an experience for people how to mitigate when the volcanic erupted without experience the real one. A game involves the characters consisting of playable character and a non-playable character. Those characters move in accordance the role given to them. This study describe a non-playable character, i.e., cows that should lead by player to a designated placed which is one of the suggestion that are directed by the national disaster management agency Indonesia. The animation that defined to the cows makes the characters have the ability to walk following the playable characters.

Key words: Non-playable character, cow animation, move following player character, game, disaster mitigation

INTRODUCTION

Indonesia is a country located on the Pacific ring of fire (Andrew, 2015) which is a place where there was a shift of the earth plates that form a line of volcanoes. This fact seated Indonesia into one of many countries that experience disasters caused by volcanic eruption. The eruption of the volcano causes material losses, environmental degradation and casualties (Anonim, 2012). Such losses could have been avoided, if the community understands how to deal with the disaster. Mitigation is one solution to respond to disasters caused by eruption, by making efforts on disaster risk reduction. Mitigation can be done structurally including physical development such as building a house with steel construction in the earthquake-prone and non-structural which is an effort to increase awareness and the ability of communities to cope with disasters such as simulating the natural disaster in disaster-prone areas (Hafidz, 2012).

Simulation is a system that is widely used and developed for reasons of cost and safety risks. Game is an attempt to make a computer simulation system more attractive (Pedersen, 2003). We are creating a simulation game that was built as a medium of learning about volcanic eruption disaster mitigation, addressed to the public so that they know what to do in a real situation.

As a game, a game simulating volcanic eruption disaster mitigation also involves characters that are divided into playable character and a non-playable character. Various missions are assigned to the playable character, one of which is a mission that requires players to interact with non-playable character. Playable character controlled by the player while non-playable character can be static or dynamic. To non-player characters sometimes is applied an artificial intelligence to take on roles as storytellers, enemies, opponents, partners or supporting. The game simulation needs non-playable character cows as supporting which part of a quest in the game. The quest is the player character should lead their cows to the area sugested. So, cows should move following the player. This study show how cows be able to walk directed by player. We implemented functions in unity to animate a non-playable character so that it responds to the playable character.

Playable and non-playable character: Characters in a game are divided into playable character and Non Playable Character (NPC). Playable character is controlled by a player while non-playable character is controlled by a program.

A non-playable character is usually used for computer animation and interactive media such as games and virtual reality and often referred to as autonomous character. There are three components of an autonomous agent (character); it has limited ability to perceive environment; it processes the information from its environment and calculates an action; it has no leader. An autonomous agent is expected to be able to coexist with the real world, so it needs to be able to learn and to keep

Corresponding Author: D.L. Crispina Pardede, Department of Computer System, Gunadarma University, Depok, Indonesia

up with the dinamically changing environment. It is also expected to be able to interact with one another (Stone, 2007).

In order to become autonomous, an agent should be able to sense it's environment and determine the response of what should be done to achieve the goals in accordance with the behavior that was given to him. The design of the behavior of the agent determined as to control the perception, selection, action, movement that make the behavior unacceptable to human reason. Behavior agent has a broad meaning, one of which behavior can be expressed as a set of actions of humans or animals based on the will or instinct.

MATERIALS AND METHODS

In a simulated environment, human players often interact with Non-player Characters (NPCs). In traditional computer games, the human user is limited by clear rules, limited role and clear objectives. Modern computer games provide a mean to entertain the human user. In that case, the ability of the game and the simulated entities to attract and keep the attention of the players determine the success. As such, there is an increasing need for control mechanisms for NPCs or "game artificial intelligence (game AI)", that produce believable and useful behaviors without detracting from the immersive experience. A common failing of traditional techniques for game AI is that the NPCs have a limited range of behaviors and apply them in limited ways. The result is that a human user can (usually) quickly determine the limits of the NPC (Kshama *et al*., 2016).

The simulation game overview: The scenario of the simulation game developed is created based on the concept of the national disaster management agency, about what should be done by residents when a volcano eruption occurs. The simulation game episode has a two directions stage. Players are required to carry out the mission given in accordance to the existing rules. In this simulation game, there is a condition of winning and losing, the player will encounter various obstacles in completing the mission such as the challenges of the time, where players will lose if they fail to complete the mission within a specified time. In some particular mission, the player also has a status bar, a stamina bar and a blood bar. If the blood in the bar runs out or reaches zero, then the player lose the game. The purpose of this simulation game is to complete the entire mission there until the end of the game. Components and conditions faced by players are given in Table 1. Some part in the scenario, playable player will interact with NPC (Kshama *et al*., 2016). This study discusses the interaction between the NPC cows with the player.

Table 1: Component and conditions of game

Components	Conditions
View point	Players have two points of view can be changed as desired from the third person and first person
Run and walk	Players can move around by walking and running. But players need stamina to run
Interaction with NPC	Players can interact with some NPCs
Reduction and charging the stamina status bar	The player's stamina bar will be reduced when player uses it to run, if the stamina bar is empty then the player can not run anymore and stamina bar will re-filled every second if the player is not running
Reduction and charging the blood status bar	The player's blood bar will be reduced when the player on barriers or obstacles in the game such as smoke or is exposed lava. The blood bar will increase every second if the stamina bar is in full condition

Fig. 1: The village map

The background place of this simulation game is a village at the foot of a mountain. The village has several areas that will be used for missions to the a variety of important elements such as mountains, fields, animal cages, housing estates, compound colony, rivers, highways, etc (Fig. 1).

The player must carry out the mission assigned to him, one of which is herding cows, in order to rescue the cow. The more animal rescued, the more rewards gained by the player. The mission of herding farm animals is referred to as 'misi4' (Johanda, 2015).

RESULTS AND DISCUSSION

Simulation game of volcanic eruption disaster mitigation is built using unity. The characters that are formed in this game are a major player as a playable

Fig. 2: The cow as NPC

Fig. 3: The player walks toward the cows

Fig. 4: The cows in idle position

character and cow as a non-playable character (Fig. 2). To gain the reward, the player must rescue the cow.

As a non-playable character, the cow is controlled by a computer program. The movements of the cow are defined to follow the player and turning to face the player. The script below is part of a program that animates the NPC cow to follow player.

Two main steps are used to make the cow walk and follow the player. The cow will move, if the player move toward the cow and reach a certain distance from the cow

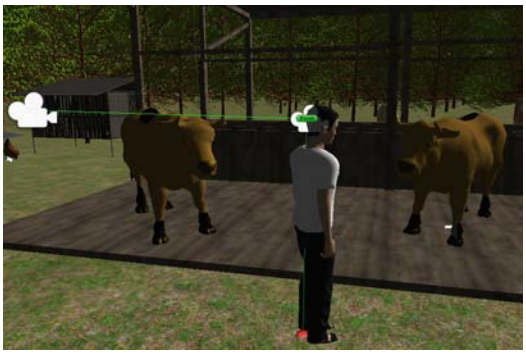

Fig. 5: The cows see the player

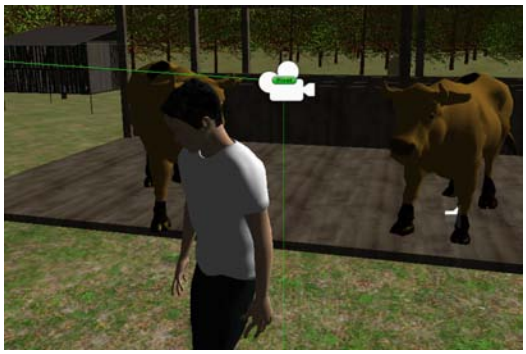

Fig. 6: The cows start to follow the player

(Fig. 3). If the distance does not meet the conditions to move, then the animal is idle (Fig. 4). We put the distance control on function update.

The distance between the player and the cow is determined by the distance between the coordinates position of NPC 'cow' (my Transform.position) and the coordinates position of the player (target.position). If the distance between the 'cow' coordinates and the player's coordinates meet some criterion, then the 'cow' will 'see' the player (Fig. 5) and move closer to the player (Fig. 6), if not, then the 'cow' will be in idle position, in other words it stops following the player. When the 'cow' in a position with it's back towards the player and the player is located near the 'cow', then the 'cow' will turn around facing the player and then following the player. We put this behavior on function jalan (Fig. 7).

The movement of the cow is based on a waypoint and the waypoint is placed on the player, resulting in motion that has the same direction as the player (Fig. 7). Various variables required are defined including the cow's speed variable (moveSpeed) from one point to another and the 'cow' turned around speed (rotationSpeed). The movement of the cow following the player is defined using the waypoint available in the unity. Detection of

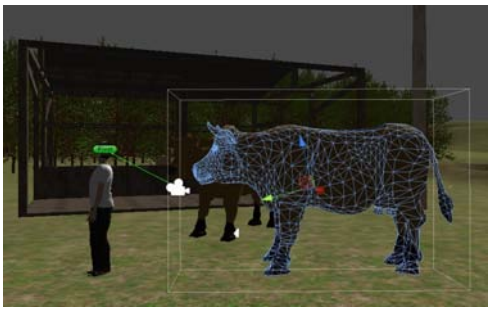

Fig. 7: The cows move in the same direction as the player

the position of the player by the NPC 'cow' is defined using a boolean type variable 'liat'. If the player is detected near the position of 'cow' (liat = true), then the 'cow' will walk following the player. If the position of the player beyond the reach of the NPC 'cow', the 'cow' will be in the 'idle' position (anim.Play ("idle")).

Algorithm:

Var target (Transform):

```
var moveSpeed = 3;
var rotationSpeed = 3;
var range : float = 10f;
var range2 : float = 10f;
var stop : float = 0;
var myTransform : Transform;
private var anim : Animator;
var waypoint : Transform[];
private var curTime : float;
var pauseDuration : float = 0;
private var character : CharacterController;
private var currentWaypoint : int = 0;
var loop : boolean = true;
var chaseSpeed : float = 3;
var liat : boolean = false;
var dampingLook = 6.0;
 function Awake()
 {
          myTransform = transform; //cache transform data for easy
access/preformance
 }
 function Start()
  {
  anim = GetComponent(Animator);
  character = GetComponent(CharacterController);
  target = GameObject.FindWithTag("Player").transform;
  }
 function jalan() {
  if (currentWaypoint < waypoint.length and liat == false){
     patrol();}
     else{
       if(loop){
       currentWaypoint=0;
       }
   }
 }
 function Update () {
//rotate to look at the player
  var distance = Vector3.Distance(myTransform.position, target.position);
     liat = false;
     if (distance<=range2 && distance>=range){
     anim.Play("idle");
     myTransform.rotation = Quaternion.SlerpmyTransform.rotation,
     Quaternion.LookRotation(target.position-
                                    myTransform.position),
rotationSpeed*Time.deltaTime);
     }
     else if(distance<=range && distance>stop){
     //move towards the player
     anim.Play("walk");
     myTransform.rotation = Quaternion.Slerp
                                    (myTransform.rotation,
     Quaternion.LookRotation(target.position-
     myTransform.position),
rotationSpeed*Time.deltaTime);
          myTransform.position += myTransform.forward*chaseSpeed*
Time.deltaTime;
     liat = true;
     }
     else if (distance<=stop) {
     anim.Play("idle");
     myTransform.rotation = Quaternion.Slerp(myTransform.rotation,
          Quaternion.LookRotation(target.position - myTransform.position),
rotationSpeed*Time.deltaTime);
     liat = true;
     }
       if (liat == false){
  jalan();
  }
 }
  function patrol () {
 //anim.Play ;
 var wp : Vector3 = waypoint[currentWaypoint].position;
 wp.y = transform.position.y;
 var moveDirection : Vector3 = wp - transform.position;
  if(moveDirection.magnitude < 0.5){
  if (curTime == 0)
   curTime = Time.time;
   anim.Play("idle");
   if ((Time.time - curTime) >= pauseDuration){
   currentWaypoint++;
   curTime = 0;
   anim.Play("walk");
   }
 }
else{
 var rotation = Quaternion.LookRotation(wp - transform.position);
    transform.rotation = Quaternion.Slerp(transform.rotation, rotation,
Time.deltaTime * rotationSpeed);
  character.Move(moveDirection.normalized*moveSpeed*Time.deltaTime);
   }
 }
```

CONCLUSION

The NPC motion animation can be done by utilizing the waypoint that is available in unity. The interaction between the playable character and the NPC is required so that a simulation game resembles the events that occur in the real world. The non-playable character 'cow' in the simulation game of volcanic eruption mitigation that was built is able to respond to the player and to move according to the role assigned to it. The NPC characters 'cow' that we created here can be used as an asset in unity so that anyone can use it in the development of other games.

REFERENCES

Andrew, T., 2015. Ring of fire. National Geo-graphic Society, Pacific Ocean. http://national geographic. org/encyclopedia/ring-fire/.

Anonim, 2012. Potential Hazards. Indonesia. http://bnpb.go.id/pengetahuan-bencana/potensi-ancaman-bencana.

Hafidz, M., 2012. Learning from Japan's disaster management. Indonesia. http://www.republika.co.id/berita/nasional/umum/12/10/10/mbnmqa-belajarmanajemen-penanganan-bencana-dari-jepang.

Johanda, M., 2015. Development of NPC Models and Behaviors for Volcanic Eruption Mitigaton Simulation Game. Gunadarma University, Depok, Indonesia,.

Kshama, T., P. Singh, S. Parmar and T. Pandey, 2016. AI & NPC in games. Int. J. Comput. Eng. Res., 3: 129-133.

Pedersen, R.E., 2003. Game Design Foundations. Wordware Publishing Inc., Texas, USA., Pages: 73.

Stone, P., 2007. Learning and multiagent reasoning for autonomous agents. Proceedings of the 20th International Joint Conference on Artifical Intelligence IJCAI 07, January 6-12, 2007, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA., pp: 13-30.