Ferran Martín Vilà

Pràctica 2: Recollida de dades

Aquest treball ha sigut realitzat a partir del joc que vam desenvolupar a projecte dos. El joc desenvolupat va ser Age of Empries II, un joc de estratègia a temps real ambientat en l 'era medieval en el que el jugador lluita contra la intel·ligència artificial per derrotar les seves unitats.

Al no tenir part online on existeixin varis usuaris, les dades analitzades son les que generen les diferents unitats que interactuen durant la partida mitjançant el combat. El jugador pot generar unitats mitjançant recursos i temps i seguidament son enviades a combatre. Les dades que es guarden serveixen tant per analitzar la producció d'unitats com el seu rendiment en combat.

Ferran Martín Vilà

Obtenció de dades:

Variables analitzades:

- EntityID: ID de la entitat que s'especifica quan es generada i s'utilitza per contar el total d'unitats generades al llarg de la partida.
- EntityType: Tipus de la unitat especificat per l'usuari.
- Name: Nom de la unitat generat aleatòriament al generar la unitat.
- Resources: Cost en recursos de la unitat.
- KillScore: Puntuació que ha aconseguit la unitat. Càlcul realitzat a partir del mal i les morts realitzades.
- TrainTime: Temps que l'usuari ha necessitat per crear la unitat.
- Kills: Morts que ha realitzat la unitat.
- DamageDone: Mal que ha realitzat la unitat.
- DamageGet: Mal que ha rebut la unitat.
- BuildingKills: Edificis que ha destruït la unitat.
- BuildingDamageDone: Mal a edificis que ha realitzat la unitat.

Per guardar aquestes variables inicialment s'ha creat una taula a MySQL amb el camps necessaris. Seguidament s'ha adaptat el codi del projecte per realitzar una crida a MySQL cada cop que una unitat del jugador es morta.

A continuació es mostra els segments de codi mes representatiu en quan a l'enviament de dades a MySQL.

Codi implementat:

```
void EntityManager::KillUnit(Unit* _unit)

{

	if(_unit->ally == true)

	{

		App->server->mysqlDatabaseGateway.SaveUnitData(_unit);

	}

	Dead_list.Add(_unit);

}

void MySqlDatabaseGateway::SaveUnitData (const Unit & _unit)
{
	DBConnection db(bufMySqlHost, bufMySqlPort, bufMySqlDatabase, bufMySqlUsername, bufMySqlPassword);

	if (!db.isConnected())return;


	DBResultSet res;

	char ID_buffer[64];
	_itoa_s(_unit.ID, ID_buffer, 10);

	char type_buffer[64];
	_itoa_s(_unit.type, type_buffer, 10);

	char resources_buffer[64];
	_itoa_s(_unit.resources, resources _buffer, 10);
```

```
	char killscore_buffer[64];
	_itoa_s(_unit.killscore, killscore _buffer, 10);

	char traint_buffer[64];
	_itoa_s(_unit.traint, traint_buffer, 10);

	char kills_buffer[64];
	_itoa_s(_unit.kills, kills_buffer, 10);

	char damage_done_buffer[64];
	_itoa_s(_unit.damage_done, damage_done_buffer, 10);

	char damage_get_buffer[64];
	_itoa_s(_unit.damage_get, damage_get_buffer, 10);

	char build_kills_buffer[64];
	_itoa_s(_unit.build_kills, build_kills_buffer, 10);

	char build_damage_buffer[64];
	_itoa_s(_unit.build_damage, build_damage_buffer, 10);

	std::string sqlStatement;

	sqlStatement =
"INSERT INTO AoE (EntityID,EntityType,Name,Resources,KillScore,TrainTime,Kills,DamageDone,DamageGet,
BuildingKills,BuildingDamageDone)
VALUES ('" + ID_buffer + "', '" + type_buffer + "', '" + _unit.name + "', '" + resources_buffer + "', '" + killscore_buffer +
"', '" + traint_buffer + "', '" + kills_buffer + "', '" + damage_done_buffer + "', '" + damage_get_buffer + "', '" +
build_kills_buffer + "', '" + build_damage_buffer + "');";

	// insert unit data
	db.sql(sqlStatement.c_str());

}
```

Ferran Martín Vilà

Resultat obtingut:

Segment de 26 mostres d'un total de 533.

Entity ID	EntityType	Name	Resources	KillScore	TrainTime	Kills	DamageDone	DamageGet	BuildingKills	BuildingDamageDone
1	3	Gabriel	97.85	1897.96	23	6	208	7	4	790
2	2	Courtney	87.84	2140.83	16	0	607	315	4	2417
3	4	Harvey	50.53	1418.27	21	1	283	1172	4	1785
4	2	Angelina	58.54	1938.38	17	2	89	275	1	829
5	0	Grace	72.15	2032.05	11	0	1086	1154	9	3299
6	1	Chad	134.21	2103.90	14	4	458	975	0	4995
7	0	Noah	73.92	1666.89	10	3	271	886	9	529
8	2	Aleksandra	54.46	2071.13	26	1	170	158	5	4358
9	1	Julius	72.62	822.18	4	1	0	557	8	238
10	0	Adalind	51.99	1911.87	3	4	537	598	6	4737
11	4	Percy	87.90	2048.69	1	1	761	813	3	3108
12	3	Davina	148.43	1542.20	12	2	72	696	1	804
13	3	Abdul	133.73	865.56	20	0	294	34	8	519
14	0	Julia	72.39	1745.71	11	0	87	171	6	262
15	2	Meredith	111.85	1576.22	22	2	27	332	3	4712
16	3	Greta	94.53	2360.14	9	5	66	493	1	3373
17	2	Hailey	79.59	1753.23	20	1	107	164	5	2042
18	3	Melody	84.12	1948.90	13	0	1087	503	6	2725
19	2	Sofia	64.40	747.44	5	4	871	1132	0	2164
20	0	Jessica	78.65	2126.93	3	3	564	1143	4	4699
21	1	Wade	132.47	1245.61	1	2	863	1141	0	3461
22	0	Peyton	140.86	1349.29	20	0	920	224	8	1580
23	3	Tyson	127.96	2054.66	13	3	175	1141	7	1500
24	2	Lara	73.32	1120.23	28	5	956	1073	5	3207
25	0	Leroy	111.46	580.99	4	5	966	750	6	4015
26	1	Peyton	107.92	1057.16	16	5	967	359	8	2498

Ferran Martín Vilà