Advanced Game Technologies coursework

Student name: Daniel Thomas Abraham

Student id: 220392033

The game consists of 4 tutorial levels and one main level

The tutorial level tells the basics of playing the game.

After the main level is finished or when player’s health reaches 0 the game ends and shows the score

Score is calculated based on the bonus captured by player and the objects destroyed

There are two types of destroyable objects: small and large cubes. Each will have its own score and hit points

GameWorld:

The world is created from the txt files in Asset/Data/

A currentScene variable helps in creating the world

GameWorld has an integer called trigger. Bit ‘0’ tells when to move to next scene. Bit ’1’ tells if the player lost. Bit ‘2’ tells if a key if found. Bit ‘3’ tells if all items are destroyed. Bit ‘4’ tells if the player won.

There are many state objects:

PlayerStateObject for Player

EnemyGameObject for Enemy

BonusStateObject for Bonuses and floor types

ObstacleStateObject for items that can be destroyed

DoorBehaviourObject for Door -> uses behaviour tree

Also there is WinStateObject which is a game object that is used as a trigger to open door or move to next level.

Each object has a layer, ignoreLayer(for ignoring collisions) and objectType( to distinguish the bonuses and check what needs to be done onCollisionBegins and onCollisionEnds).

PhysicsSystem:

The game runs with basic collision detection by default but can use quad tree by pressing ‘B’ key.

AngularImpulse and AngularVelocity added.

AABB and sphere volumes are implemented

OBB interacts only with sphere(can be seen in walls of first tutorial level)

Penalty method is used for floors with object interactions where the floors appear in magenta in the game world.

Friction and elasticity are implemented for objects based on which impulse calculations occur

Player:

Player is controlled by WASD, Space for jump and mouse for orientation. Player is moved using force

Player has a state machine. If idle(no wasd or space input from user) the player will start rotating

Player can interact with many objects in the world.(PlayerStateObject)

Player cannot jump again based on raycast on to the floor(TutorialGame)

Game ends when player finishes all levels or player’s health reaches 0. (TutorialGame)

Enemy:

The enemies are state machines which have two behaviours: They patrol about a given area and if the player is close to them they will run away. (EnemyGameObject)

Enemies can get bonuses but can also set their ignoreLayer to not collide with bonuses

Enemies can move towards the player(EnemyGameObject and StateGameObject)

Enemies can collide with obstacles and destroy them.

Door:

Door hinders the player from finishing the level. Door opens based on two triggers :No items remain or key is found

Door uses behaviour trees(DoorBehaviourObject)

Door opens based on creation of constraint between the visible box and invisible physics game object which is above the visible box.

Door closes by removing the constraint

Can test the functionality opening debug mode(Press F6) and pressing 3(sets trigger to 0), 4(sets trigger to 4) and 5(sets trigger to 8)

Levels:

Levels are generated based on txt files.

Each character creates one of the floor types:(TutorialGame)

x -> wall

s-> secret wall which player can go through but not other objects

.-> floor

p->player

e->enemy

b->speed bonus(adds score and increases the speed of the player)

l->lava floor (reduces the players health by specific amount on collision)

h->healing floor(increases player’s health by specific amount on collision)

q->small square object

a->large square object

m->mud floor(slows down player)

i->ice floor(speeds up player)

d->door

k->key

w->win area

j->secret wall with key

r->spring floor(for penalty method calculation)

Grid based Pathfinding implemented

Basic Networking to send and receive packets by server and client

Menus implemented to start the game as well as when player wins/loses

Keyboard actions:

WASD, Space -> moves player

Mouse-> orients camera and player

F3 changes control to move camera: WASD, Shift, Space ->Moves camera

F1 reinitializes the level

F2 reinitializes camera (when controlling camera)

F6 enable debug mode

In debug mode :

7,8,9,0 -> changes gravity

+,- -> changes damping

Q->selection mode to select and add forces. L->locks object that is selected