Indrei Valentina Andreea

WomXn Develop at Ubisoft

Documentation for the game

The plot of the game is inspired after greek mythology. The player is controlling Nyx, the goddess of the night and fights the three main gods of Olympus (Poseidon, Hades, Zeus). Nyx's allies are the other gods which seek to overthrow the three brothers. The goal is to survive as long as possible and to accumulate points.

```
137
138  void updateLeaderBoard() { ... }
188
189  void environmentMovement() { ... }
245
246  void healthDropEvent(int al) { ... }
279
280  void heartCollision() { ... }
291
292  void rockCollision() { ... }
309
310  void monsterCollision() { ... }
339
340  bool damageCooldown(bool damageFlag) { ... }
353
354  void kickF() { ... }
364
365  void jumpF() { ... }
381
382  void checkAresPowerup() { ... }
481
482  void checkApolloPowerup() { ... }
515
516  void teleport() { ... }
634
635  void obstacleSpawner() { ... }
```

The main functions used

Basic actions

- **Jump**

  To jump the player presses the gamepad UP button. The player needs to jump in order to not take damage from incoming rocks or to collect hearts. The jump is vertical and it avoids collision.

```
void jumpF()
{
    state.jump_dur++;
    float x, y;
    testSprite->GetPosition(x, y);
    if (state.jump_dur > 20)
        y = y - 10;//+-10 este in ex asta cat de mult se sare
    else
        y = y + 10;
    testSprite->SetPosition(x, y);
    if (state.jump_dur == 40) // k e cat de mult dureaza saritura
    {
        state.jump_dur = 0;
        state.jump_restriction = 5; //k2 intervalul dintre doua sarituri
    }
}
```

- **Attack**

  The attack function is executed by pressing the A button. One needs to attack in order to defeat the cerberus (hitting it is the only way to avoid it). Attacking spins the avatar and instead of taking damage on collision with a cerberus it kills it, triggering an animation

```
void kickF()
{
    kick.jump_dur++;
    testSprite->SetAngle(testSprite->GetAngle() - 0.25132f);
    if (kick.jump_dur == 25) // k e cat de mult dureaza saritura
    {
        kick.jump_dur = 0;
        kick.jump_restriction = 5; //k2 intervalul dintre doua sarituri
    }
}
```

- **Teleport to shadow**

  Each object (rock or cerberus) spawns with a shadow underneath. The shadow can be of 3 types (pointing to the right, to the center or to the left). The player can teleport to any of these (by pressing B) but only in some cases it is to the player advantage.

If the shadow is pointing to the left you take damage regardless of wether the object is a cerberus or a rock (with the exception that in the case of the cerberus one can quickly attack after teleportation and avoid damage).

If the shadow is central then the player takes damage from the cerberus but not the rock.

If the shadow points to the right the player doesn't take damage.

Indrei Valentina Andreea

Obstacles/enemies

- **Rock**
  The rock is an obstacle that the player needs to avoid by jumping over it.
- **Cerberus**
  The cerberus is a creature that attacks the player unless it is attacked first. (cannot be avoided by jumping)

Detecting collisions

- **Rock**
  Detects collision unless the player jumps.
- **Cerberus**
  Detects collision unless the player avatar is performing a kick (an attack).
  Collision is detected by checking how close the coordinates of the object and player are.

Health system

```
void healthDropEvent(int al)
{
    if (al == 0)
        nr_hearts--;
    else
        if(nr_hearts<=5 && al==1)
            nr_hearts++;
    switch (nr_hearts)
    {
    case 6: healthSprite->SetAnimation(ANIM_HEALTH6);
        break;
    case 5: healthSprite->SetAnimation(ANIM_HEALTH5);
        break;
    case 4: healthSprite->SetAnimation(ANIM_HEALTH4);
        break;
    case 3: healthSprite->SetAnimation(ANIM_HEALTH3);
        break;
    case 2: healthSprite->SetAnimation(ANIM_HEALTH2);
        break;
    case 1: healthSprite->SetAnimation(ANIM_HEALTH1);
        break;
    case 0:
    {
        delete healthSprite;
        nohealthSprite->SetPosition(105.0f, 735.0f);
        gs = inLeader;
        leaderflag = 1;
        break;
    }
    default: //nohealthSprite->SetPosition(105.0f, 735.0f);
        break;
    }
}
```

The player starts with 6 health points which can be lost in a collision with a hostile object or gained if a heart is collected.

End game condition

The game ends when the player's health points reach 0.

Main menu

```
void Update(float deltaTime)
{

    switch (gs)
    {
    case 0:
    { ... }
    case 1:
    { ... }
    case 2:
    { ... }
    case 3:
    { ... }
    case 4:
    { ... }
    case 5:
    { ... }
    case 6:
    { ... }
    default: { ... }
    }
}
```

```
enum GameState
{
    inMenu,
    inGame,
    inMyth,
    inStory,
    inLeader,
    inGuide,
    pause
};
```

When you open the game the first thing you see is the menu which has 5 options:

- **Start**
- **Greek mythology**
- **Story line**
- **Leaderboard**
- **Controller guide**

Indrei Valentina Andreea

Menu to support start, pause, resume or restart actions

- **Start**
  Start appears in the main menu
- **Pause**
  When in game one can press the start button to access the pause menu in which the player can choose to resume the game or restart it.

Scrolling background

```
    backgroundSprite1->GetPosition(xb1, yb1);
    xb1 -= 3.0f;
    backgroundSprite2->GetPosition(xb2, yb2);
    xb2 -= 3.0f;
    groundSprite->GetPosition(xg1, yg1);
    groundSprite2->GetPosition(xg2, yg2);
    xg1 -= 3.0f;
    xg2 -= 3.0f;
    groundSprite->SetPosition(xg1, yg1);
    groundSprite2->SetPosition(xg2, yg2);
    backgroundSprite2->SetPosition(xb2, yb2);
    backgroundSprite1->SetPosition(xb1, yb1);
    if (xg1 <= -509.0f)
    {
        groundSprite->SetPosition(1536.0f, 100.0f);
        groundSprite2->SetPosition(512.0f, 100.0f);
    }
    if (xg2 <= -509.0f)
    {
        groundSprite2->SetPosition(1536.0f, 100.0f);
        groundSprite->SetPosition(512.0f, 100.0f);
    }
    if (xb1 <= -509.0f)
    {
        backgroundSprite1->SetPosition(1536.0f, 483.5f);
        backgroundSprite2->SetPosition(512.0f, 483.5f);
    }
    if (xb2 <= -509.0f)
    {
        backgroundSprite2->SetPosition(1536.0f, 483.5f);
        backgroundSprite1->SetPosition(512.0f, 483.5f);
    }
}
```

- **Background**
  In order to create the scrolling background I used 2 photos that are next to each other and move together to the left of the screen.
- **Ground**
  Same for the ground.

Indrei Valentina Andreea

Heads-up display

- **Avatar's health**
- **Apollo progress**
- **Ares progress**
- **Game time**
- **Score**

Collectibles

- **Hearts**
  Hearts are objects that spawn rarely and when caught (by jumping toward them) restore health points.

Power-ups

There are 2 power-ups the player can use when he/she accumulates enough points. For each of them 10 points are required.

The points that are specific to Apollo are gained for using the teleport abillity.

The points for Ares are gained for killing Cerberus.

```
void checkAresPowerup() { ... }

void checkApolloPowerup()
{
    if (powerup_Apollo == 10)
    {
        apollo_avSprite->SetPosition(85.0f, 650.0f);
        apolloSprite->SetPosition(-500.0f, 650.0f);
    }

    if (App::GetController().CheckButton(XINPUT_GAMEPAD_LEFT_SHOULDER, true) && powerup_Apollo == 10)
    {
        powerup_Apollo = 0;
        apolloSprite->SetPosition(85.0f, 650.0f);
        apollo_avSprite->SetPosition(-500.0f, 650.0f);
        suntimer = 500;
        sunSprite->SetPosition(-50, 530.0f);
        customScore = customScore + 30;
    }

    sunSprite->GetPosition(xo, yo);
    if (suntimer != 0)
        suntimer = suntimer - 1;
    else sunSprite->SetPosition(-500.0f, 530.0f);
    if (suntimer > 0 && suntimer % 5 == 0)
    {
        sunSprite->GetPosition(xb1, yb1);
        if (suntimer > 250)
            xb1 = xb1 + 5;
        else
            xb1 = xb1 - 5;
        sunSprite->SetPosition(xb1, 530.0f);
    }

}
```

```
if (suntimer > 0)
    shadowType = 0;
switch (shadowType)
{
case 0:
    shadows[shadow1Counter]->SetPosition(1500.0f, 150.0f);
    shadow1Counter++;
    break;
case 1:
    shadows[shadow2Counter]->SetPosition(1500.0f, 150.0f);
    shadow2Counter++;
    break;
case 2:
    shadows[shadow3Counter]->SetPosition(1500.0f, 150.0f);
    shadow3Counter++;
    break;
}
obstacleSpawned = 1;//flag for if object is spawned
```

- Apollo

After calling Apollo, all shadows point to the right for a brief period of time.

- Ares

After calling Ares, a sword falls from the sky and prevents any obstacles from reaching the player (effectively granting invulnerability).

Indrei Valentina Andreea

Scoring system

- **Game time**
- **Custom scoring system**
  Score = Game time + 30 per used powerup + 10 for each use of the shadow + 5 for each Cerberus kill

Leaderboard

In order to store the 5 highest scores I used a local file. The program can modify the file accordingly and can also clear it with the press of a button.

```
case 4:
{
    leaderSprite->Draw();
    std::string ld = std::to_string((int)customScore);
    char const* leader_pchar = ld.c_str();
    App::Print(430, 395, "Ai terminat cu scorul: ", 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_12);
    App::Print(550, 395, leader_pchar, 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_12);
    ifstream fin("leaderfile.txt");
    for (int i = 0; i < 5; i++)
    {
        fin >> score[i];
    }
    if (leaderflag == 1 )
    {
        updateLeaderBoard();
    }
    if(congratflag == 1)
        App::Print(420, 330, "Felicitari, ai ajuns in leaderboard!", 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_12);

    for (int i = 0; i < 5; i++)
    {
        std::string ld = std::to_string(score[i]);
        char const* leader_pchar = ld.c_str();
        App::Print(490, i * 50 + 100, "Locul", 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_18);
        //std::string ldi = std::to_string(i);
        //char const* i_pchar = ldi.c_str();
        //App::Print(540, i * 50 + 100, i_pchar, 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_18);
        App::Print(512, i * 50 + 80, leader_pchar, 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_18);
    }
    int kx = 1;
    for (int i = 4; i >= 0; i--)
    {

        std::string ldi = std::to_string(kx);
        char const* i_pchar = ldi.c_str();
        App::Print(540, i * 50 + 100, i_pchar, 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_18);
        kx++;
    }
    App::Print(425, 50, "Press A to reset the leaderboard", 1.0f, 1.0f, 1.0f, GLUT_BITMAP_HELVETICA_12);
    if (App::GetController().CheckButton(XINPUT_GAMEPAD_A, true))
    {
        for (int i = 0; i < 5; i++)
        {
            score[i] = 0;
            ofstream fout("leaderfile.txt");
            for (int i = 0; i < 5; i++)
            {
                fout << score[i] << endl;
            }
        }
    }
    App::Print(750, 20, "Press B to go back to Menu ");
    break;
```