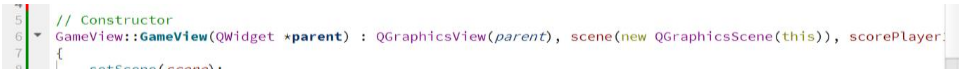
CENG 206 Project 2 – Hide-and-seek Game

Group 9

Project Group Participants:

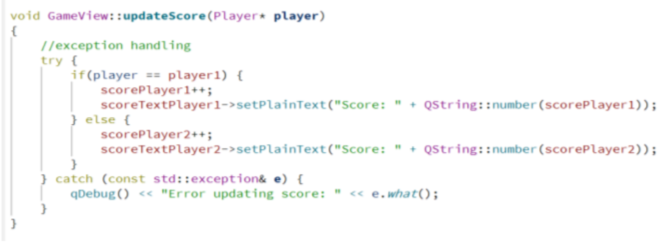
- Beyza Nur Deliktaş 21050111008
- İrem Şahar 21050141024
- Osmancan Çağlayan 21050111074

a. Inheritance



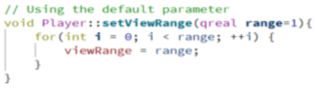
```
// Constructor
GameView::GameView(QWidget *parent) : QGraphicsView(parent), scene(new QGraphicsScene(this)), scorePlayer
{
    setScene(scene);
```

b. Exception handling



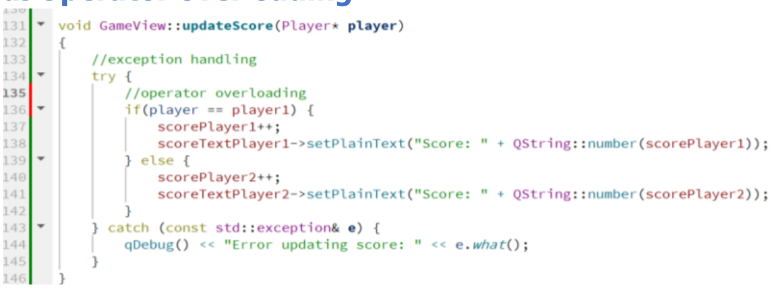
```
void GameView::updateScore(Player* player)
{
    //exception handling
    try {
        if(player == player1) {
            scorePlayer1++;
            scoreTextPlayer1->setPlainText("Score: " + QString::number(scorePlayer1));
        } else {
            scorePlayer2++;
            scoreTextPlayer2->setPlainText("Score: " + QString::number(scorePlayer2));
        }
    } catch (const std::exception& e) {
        qDebug() << "Error updating score: " << e.what();
    }
}
```

c. Default parameter



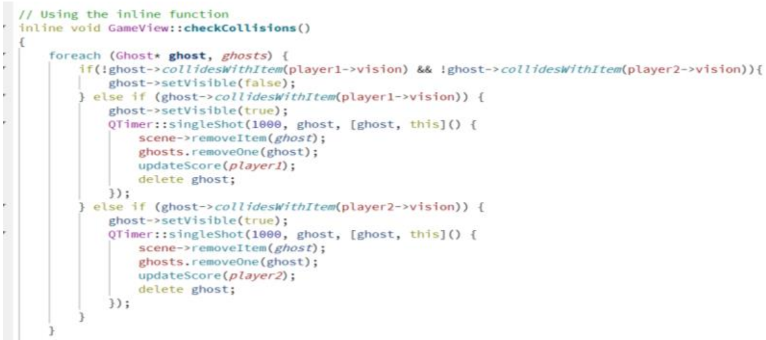
```
// Using the default parameter
void Player::setViewRange(qreal range=1){
    for(int i = 0; i < range; ++i) {
        viewRange = range;
    }
}
```

d. Operator overloading



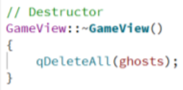
```
void GameView::updateScore(Player* player)
{
    //exception handling
    try {
        //operator overloading
        if(player == player1) {
            scorePlayer1++;
            scoreTextPlayer1->setPlainText("Score: " + QString::number(scorePlayer1));
        } else {
            scorePlayer2++;
            scoreTextPlayer2->setPlainText("Score: " + QString::number(scorePlayer2));
        }
    } catch (const std::exception& e) {
        qDebug() << "Error updating score: " << e.what();
    }
}
```

e. Inline function

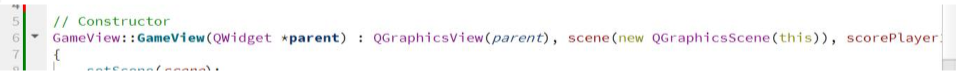


```
// Using the inline function
inline void GameView::checkCollisions()
{
    foreach (Ghost* ghost, ghosts) {
        if(!ghost->collidesWithItem(player1->vision) && !ghost->collidesWithItem(player2->vision)){
            ghost->setVisible(false);
        } else if (ghost->collidesWithItem(player1->vision)) {
            ghost->setVisible(true);
            QTimer::singleShot(1000, ghost, [ghost, this]() {
                scene->removeItem(ghost);
                ghosts.removeOne(ghost);
                updateScore(player1);
                delete ghost;
            });
        } else if (ghost->collidesWithItem(player2->vision)) {
            ghost->setVisible(true);
            QTimer::singleShot(1000, ghost, [ghost, this]() {
                scene->removeItem(ghost);
                ghosts.removeOne(ghost);
                updateScore(player2);
                delete ghost;
            });
        }
    }
```

f. Constructor/destructor

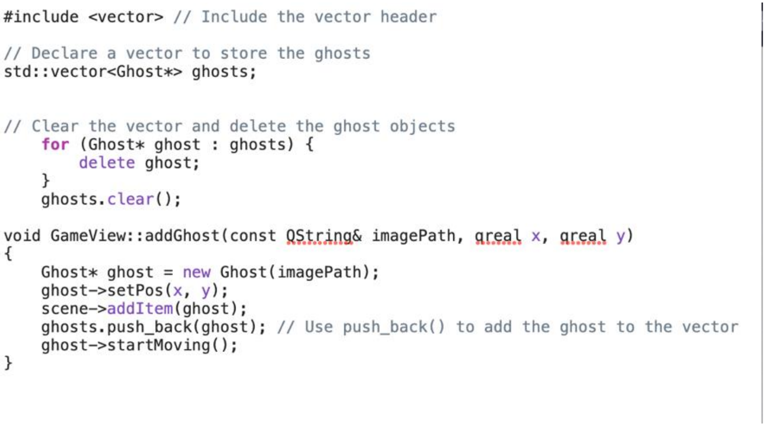


```
// Destructor
GameView::~GameView()
{
    qDeleteAll(ghosts);
}
```



```
// Constructor
GameView::GameView(QWidget *parent) : QGraphicsView(parent), scene(new QGraphicsScene(this)), scorePlayer
{
    setScene(scene);
```

g. Vectors



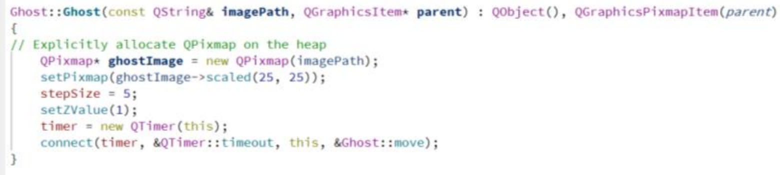
```
#include <vector> // Include the vector header

// Declare a vector to store the ghosts
std::vector<Ghost*> ghosts;


// Clear the vector and delete the ghost objects
    for (Ghost* ghost : ghosts) {
        delete ghost;
    }
    ghosts.clear();

void GameView::addGhost(const QString& imagePath, qreal x, qreal y)
{
    Ghost* ghost = new Ghost(imagePath);
    ghost->setPos(x, y);
    scene->addItem(ghost);
    ghosts.push_back(ghost); // Use push_back() to add the ghost to the vector
    ghost->startMoving();
}
```

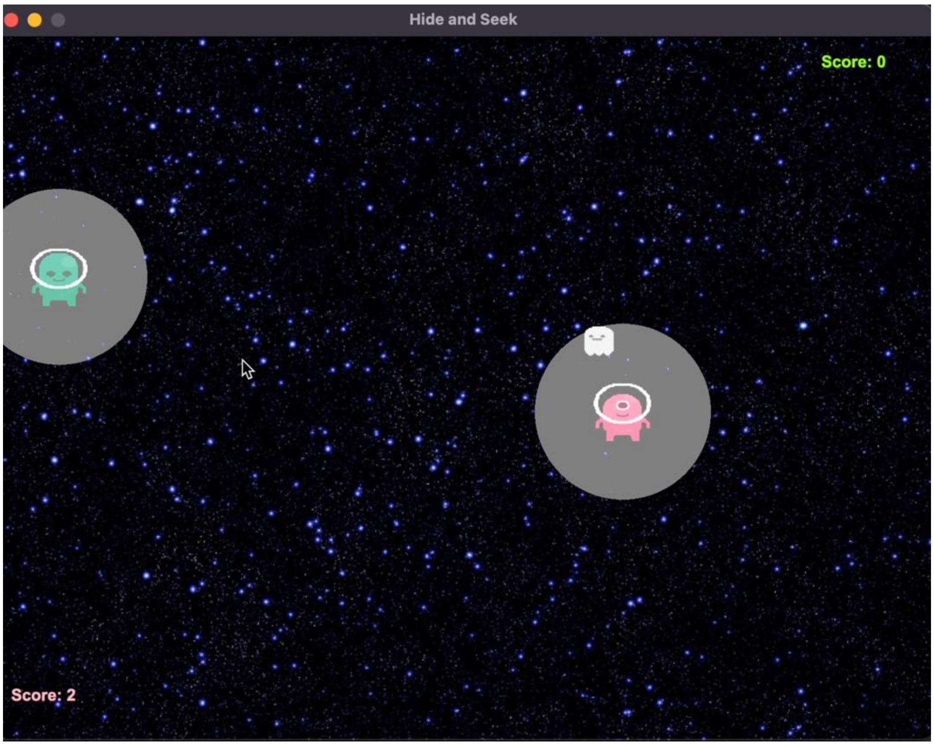
h. Explicit heap dynamic variable

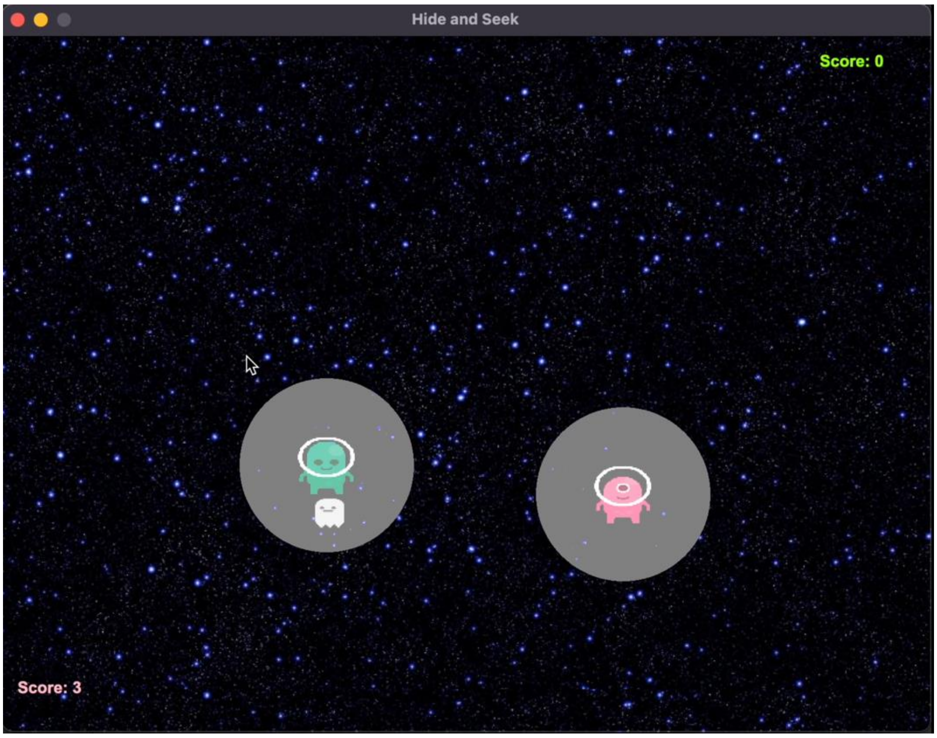


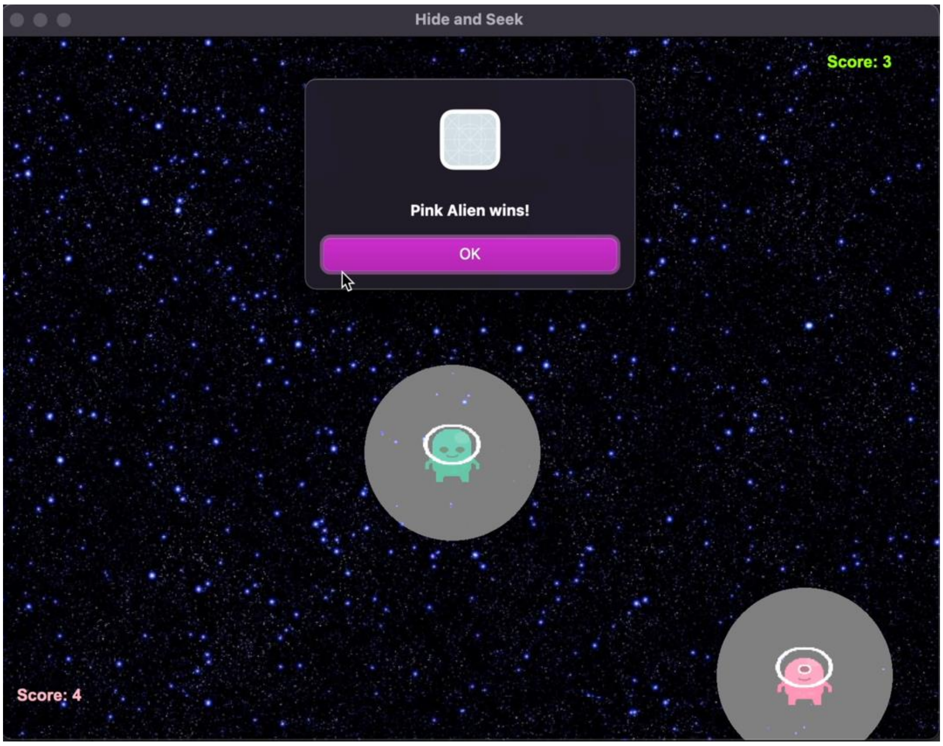
```
Ghost::Ghost(const QString& imagePath, QGraphicsItem* parent) : QObject(), QGraphicsPixmapItem(parent)
{
// Explicitly allocate QPixmap on the heap
    QPixmap* ghostImage = new QPixmap(imagePath);
    setPixmap(ghostImage->scaled(25, 25));
    stepSize = 5;
    setZValue(1);
    timer = new QTimer(this);
    connect(timer, &QTimer::timeout, this, &Ghost::move);
}
```

Screenshots of the Game

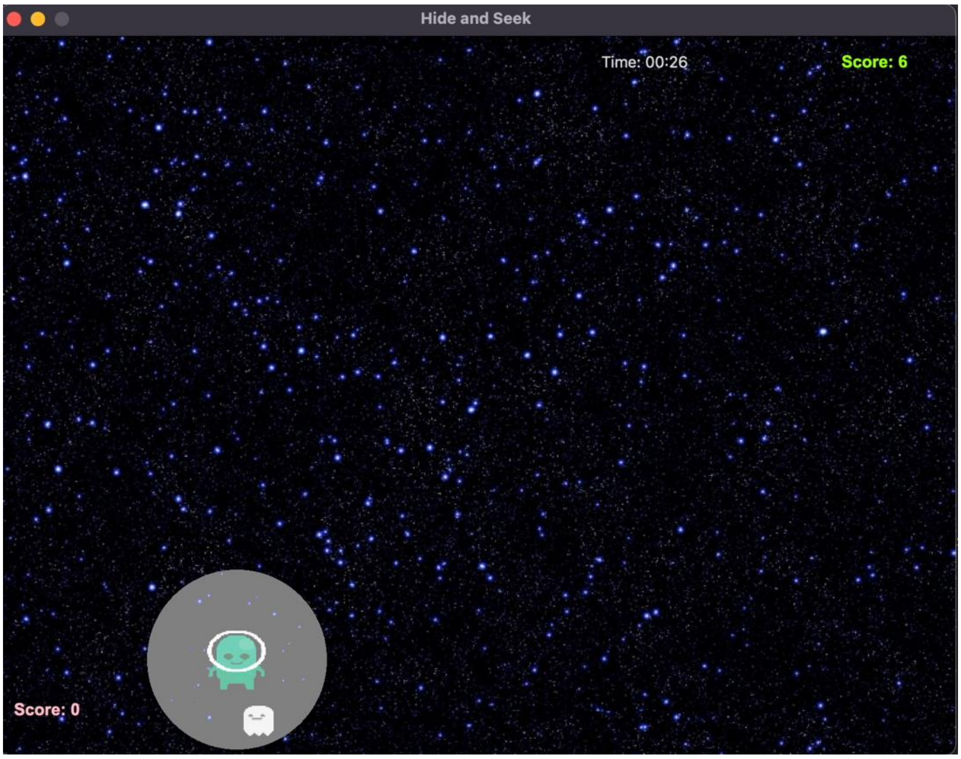
As mentioned in the presentation and on github, there are 2 versions of our game (the code for each version is available on github, also the gameplay video of version 2 is available too). However, in the 2nd version, since the problem of not opening on some computers was encountered, we presented the safer one as the main game in our project.

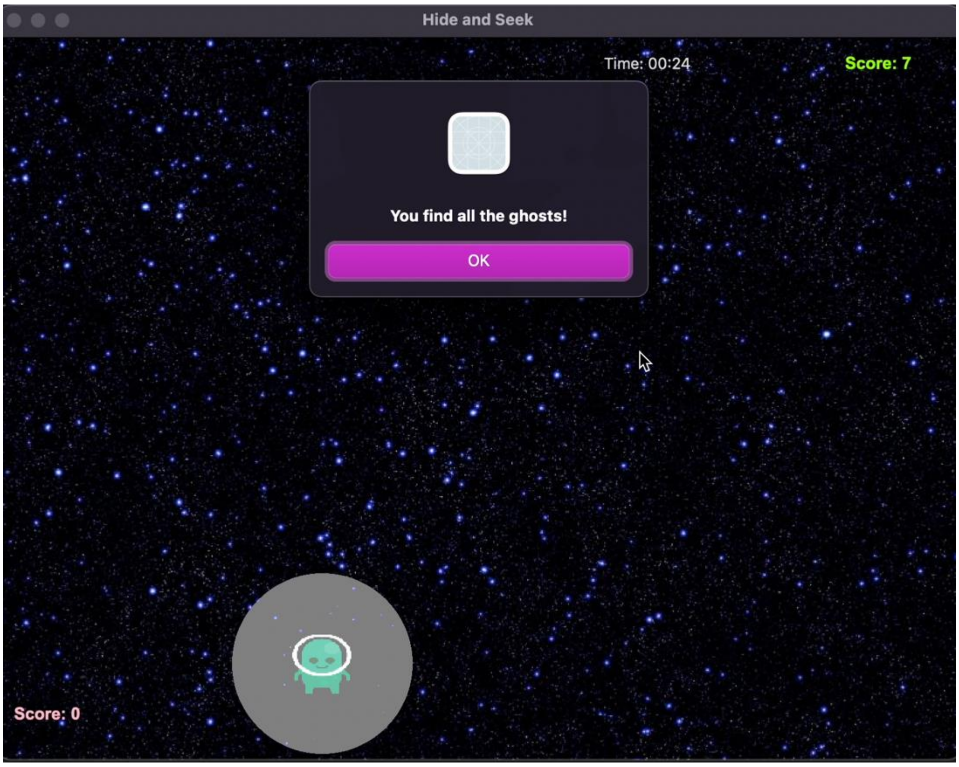




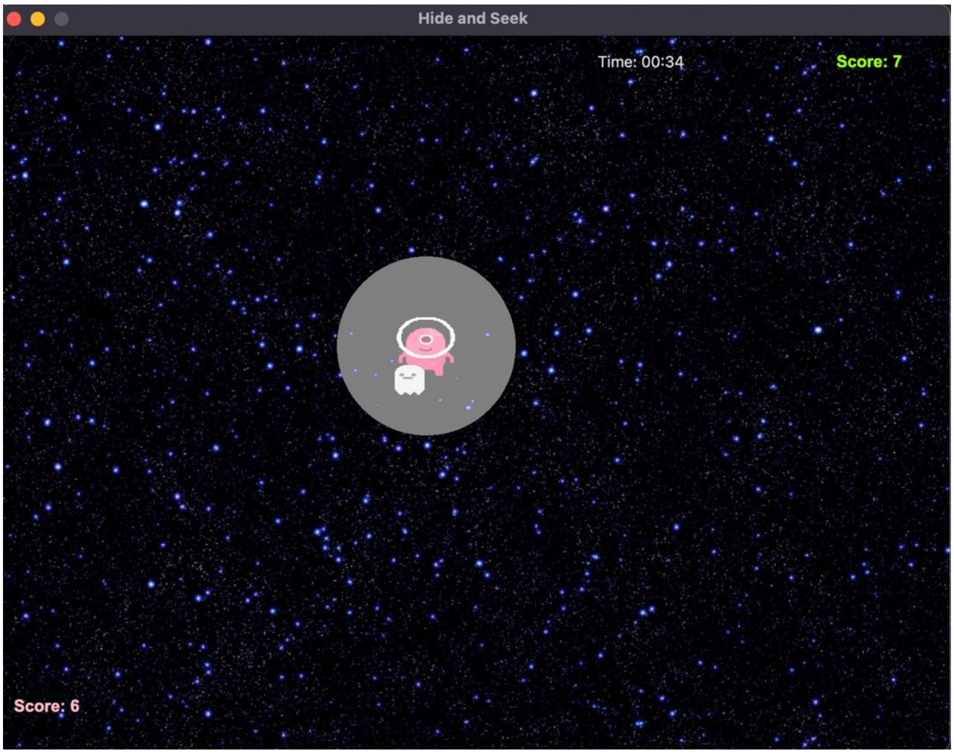


In version 2, there is a game with 2 rounds and a timer. The player who catches the most ghosts during the time is the winner.

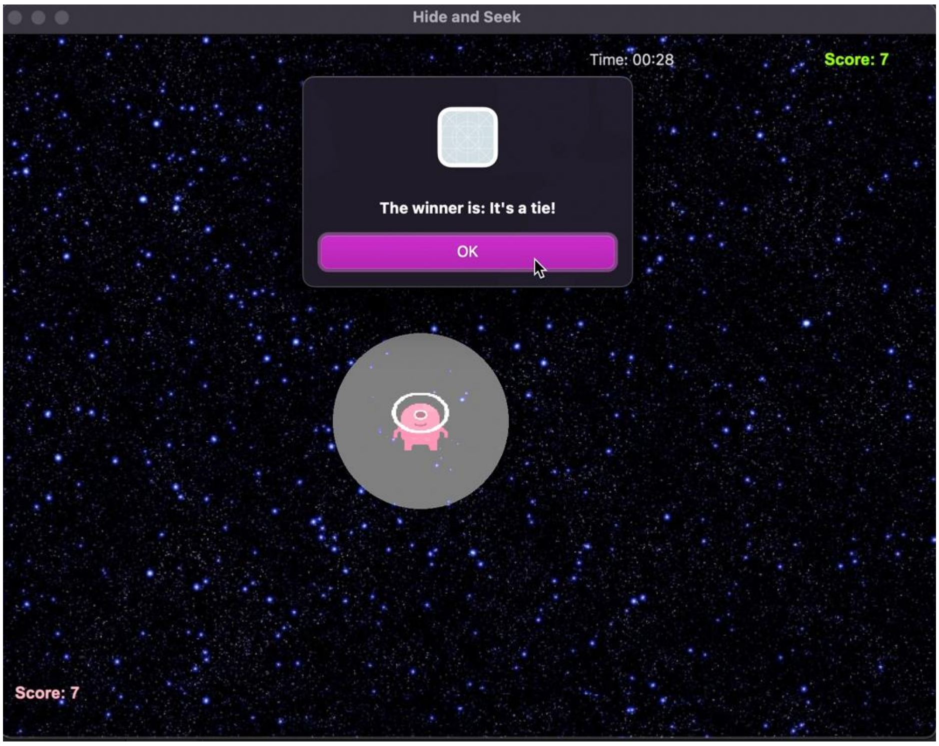




Hide and Seek
Time: 00:24
Score: 7
You find all the ghosts!
OK
Score: 0



Hide and Seek
Time: 00:34
Score: 7
Score: 6



The winner is: It's a tie!