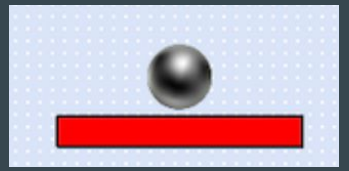
Ice Breaker

Olivia Heiner, Jeremy Toop, Danya Elgebaly

Basic Overview

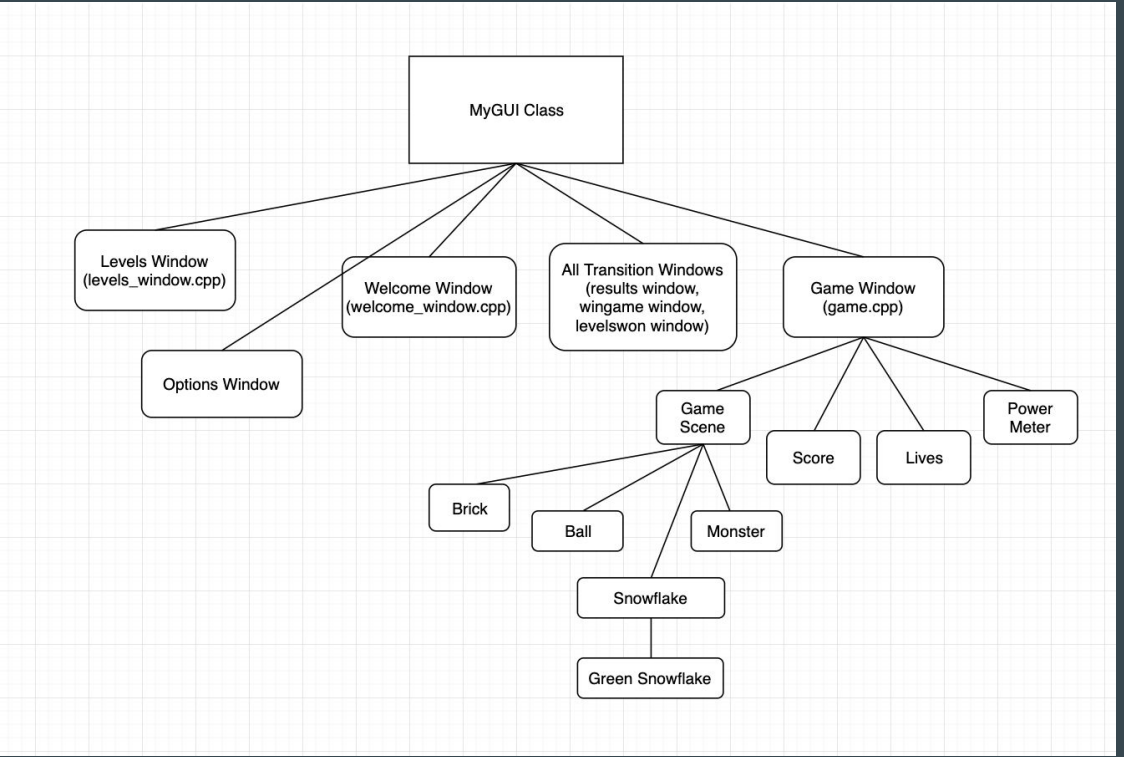
- Bounce ball to break bricks
- Use arrow keys on laptop to move paddle left and right
- Power-ups
  - Various power-ups to help you get through the game
  - Hit monsters to get closer toward gaining a powerup
  - Snowflakes - catch them to gain xp for getting powerups
  - Green Snowflakes - catch them to get lives



Class Hierarchy

Classes

- MyGUI, Levels Window, Welcome Window, All transition Windows, Game Window
- Main.cpp creates an object from the myGUI class which creates all of the windows and manages connections

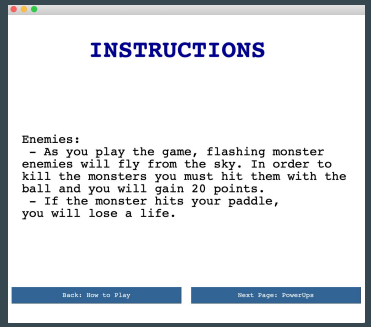


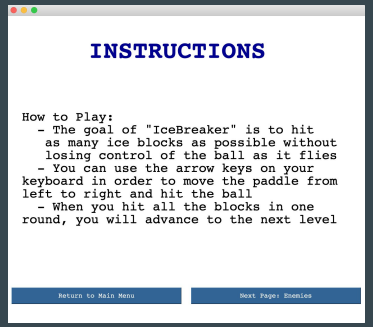
Memory Management

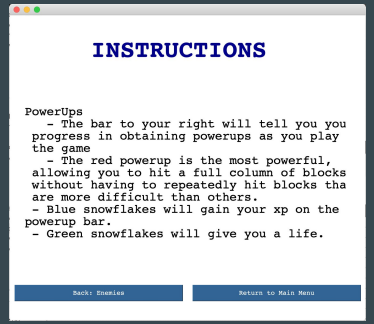
- All windows are initialized when the program begins
- The game view (within the game window) is the only thing that is not initialized until the game is started. It is initialized based on the level.
- In between levels, bricks, paddles, balls, and any other objects within the game are deleted and then replaced with new ones for the next level. Score and lives carry over to the next level.
- If the game is quit, then the score and lives are all reset.
- All objects that are temporarily in the game, such as falling snowflakes, monsters, and powerup items are deleted when they leave the screen.

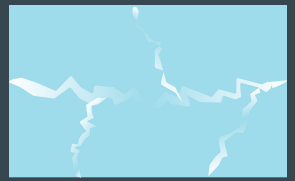
UI Design Components

- Welcome window, instructions window, transitioning windows
- UI Design file and Stacked Widget
- Stylesheets
- External Design
  - Illustrator
  - Online paint tools









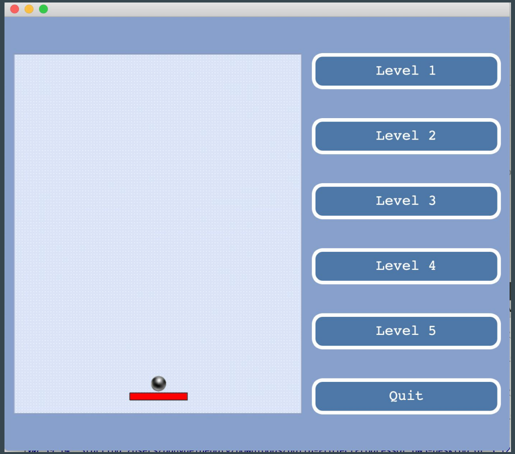
Collision Detection

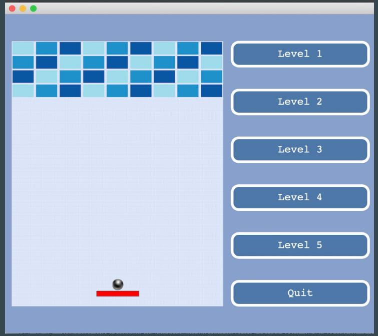
```
//Check for collisions with items
QList<QGraphicsItem*> items_ball_hit = collidingItems();
for(int i=0;i<items_ball_hit.size();i++)
{
    //Check if the collision was with a brick
    if(typeid(*items_ball_hit[i])==typeid(brick))
    {
        brick* hitbrick = dynamic_cast<brick*>(items_ball_hit[i]);
        if(is_powered_ball)
        {
            player->play();
            hitbrick->destroy_brick();
        }
        else
        {
             //Check if brick was hit on a right or left side, then change x velocity
             if(getMiddleXCoord() < hitbrick->x() && y() < hitbrick->getBottomY()-5 &&
                     y()+boundingRect().height()*scale > hitbrick->y()+5 && x_velocity>0)
                 x_velocity = -abs(x_velocity);
             else if(getMiddleXCoord()>hitbrick->getRightX() && y() < hitbrick->getBottomY()-5 &&
                     y()+boundingRect().height()*scale > hitbrick->y()+5  && x_velocity<0)
                 x_velocity = abs(x_velocity);

             //Check if brick was hit on a bottom or top side, then change y velocity
             else if(y()<hitbrick->y()&&y_velocity>0)
                 y_velocity = -abs(y_velocity);
             else if(y()>hitbrick->y()&&y_velocity<0)
                 y_velocity = abs(y_velocity);
             player->play();
             hitbrick->update_hit_brick();
        }

    }
```

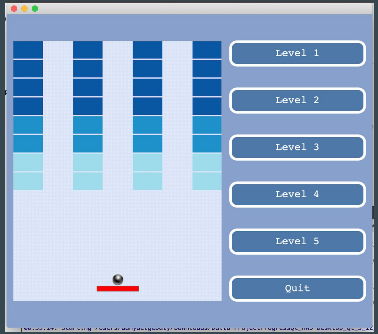
Collision Detection

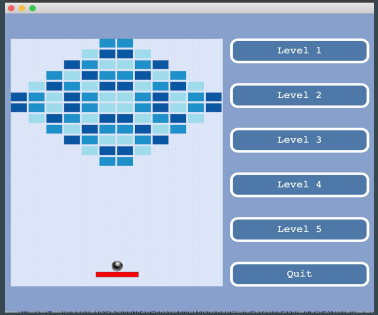
```
//Check if a block isn't hit, but instead a paddle is hit
else if(typeid(*items_ball_hit[i])==typeid(paddle))                    ⚠ expression wi
{
    paddle* hitpaddle = dynamic_cast<paddle*>(items_ball_hit[i]);
    double reflection_angle = 0;
    double ball_pos = this->getMiddleXCoord();
    double paddle_pos = hitpaddle->getMiddleXCoord();
    if(ball_pos < paddle_pos)
    {
        reflection_angle = 100+(140-100)*(paddle_pos-ball_pos)/((hitpaddle->getwidth()+ballWidth/2)/2);
        x_velocity = ball_speed*qCos(qDegreesToRadians((reflection_angle)));
    }
    else
    {
        reflection_angle = 80-(80-40)*(ball_pos-paddle_pos)/((hitpaddle->getwidth()+ballWidth/2)/2);
        x_velocity = ball_speed*qCos(qDegreesToRadians((reflection_angle)));
    }
    y_velocity = -ball_speed*qSin(qDegreesToRadians(static_cast<double>(reflection_angle)));
    player->play();
}
```

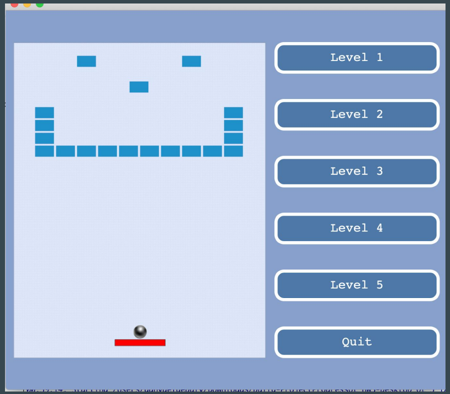
DEMO

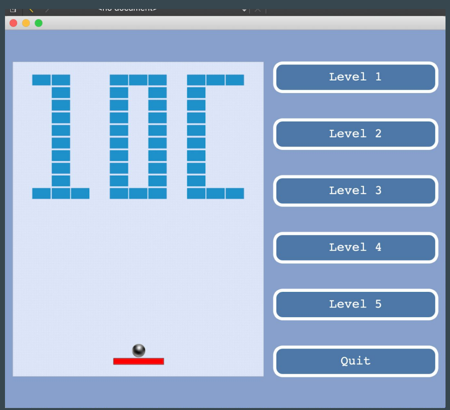

Level 1
Level 2
Level 3
Level 4
Level 5
Quit


Level 1
Level 2
Level 3
Level 4
Level 5
Quit


Level 1
Level 2
Level 3
Level 4
Level 5
Quit


Level 1
Level 2
Level 3
Level 4
Level 5
Quit


Level 1
Level 2
Level 3
Level 4
Level 5
Quit


Level 1
Level 2
Level 3
Level 4
Level 5
Quit