Chad Martin
CSCI-102
Fall 2012

Homework 2 Design Document

Purpose/Overview

The program starts out by displaying intro screen 1 which asks the user for a number from 1 to 5 which correlates to the speed of the snake and the points multiplier. 1 is the slowest speed and 5 is the fastest speed. The faster the speed the bigger the points multiplier.

Next intro screen 2 is displayed, the user is asked if he/she would like to enable wrap around. If wrap around is enabled, the snake is allowed to go through the borders. If wrap around is disabled and the snake hits the borders, it will cause a game over. Also, If wrap around is disabled, the points multiplier is doubled.

After the game options are set, the game starts at 0 points with 2 pieces on the screen. One of the pieces is a SnakeBlock head. This piece is the snake that the user controls with the arrow keys. The other piece is a green Apple logo. This piece is a normal piece of food. When the snake collides with the normal piece of food, the snake will grow 1 block and the points will increase by 7 times the points multiplier. After the collision, the food will spawn in a new random location. Also in the window, there is an iPhone. This is the border around the screen. On the status bar of the iPhone, the points that the user has accumulated are displayed. If the snake head runs over another part of the snake, it will always cause a game over. If wrap around is disabled and the snake hits the border, it will cause a game over.

Another piece that can spawn is a Windows 8 logo. This piece is a special piece of food. When the snake collides with the special piece of food, the game will go into a blue screen of death. This piece will spawn every ten grows or if ‘W’ is pressed.

Another piece that can spawn is a Linux logo. This piece is the winning piece of food. When the snake collides with the winning piece of food, the game will end and a victory screen will appear. This piece will spawn after the special piece of food is painted in 100 paint events or if ‘L’ is pressed.

When a blue screen of death occurs, the game stops, the users points are set to zero, and a blue screen of death is displayed. After any key is pressed the game moves on to the game over screen.

When a game over occurs, the game stops and the game over screen is displayed. On the game over screen, the words game over, the total number of points, and instructions for the user to play again are displayed. If the user presses the Space bar, the game starts over and intro screen 1 is displayed.

When a victory occurs, the game stops and the victory screen is displayed. On the victory screen, the words victory, the total number of points, and instructions for the user to play again are displayed. If the user presses the Space bar, the game starts over and intro screen 1 is displayed.

If the user presses ESC at any time during the program, the program will quit.

Requirements

Entertain the user through playing the game. Find the Linux logo and eat it to win the game.

Instructions

In order to compile this program,

- extract files from ChadMart_HW2.zip
- navigate to the `/HW2/ directory
- enter “qmake -project” in terminal
- enter “qmake” in terminal
- enter “make” in terminal

In order to run this program,

- enter “./hw2” in terminal

Program library dependencies

- QT library
  - <QDesktopWidget>
  - <QApplication>
  - <QWidget>
  - <QKeyEvent>
  - <QImage>
  - <QRect>
  - <QPainter>
- C Standard General Utilities Library
  - <stdlib.h>
- vector Library
  - <vector>

Game Class Reference

The **Snake Game** Logic. This class contains the logic that makes up the **Snake Game**. More...

`#include <Game.h>`

List of all members.

Public Member Functions

	Game (QWidget *parent=0) Class Constructor - initializes the data members to default values and creates dynamically allocated memory. Precondition: **Game** object has been created. Postcondition: Intro screen 1 is displayed.
	~Game () Class Destructor - Deletes dynamically allocated objects (specialFood, foodWinner, food, and snake).

Protected Member Functions

void	**paintEvent** (QPaintEvent *event) Paints the necessary object on the screen. Precondition: a QPaintEvent is called. Postcondition: Necessary objects are painted on the screen.
void	**timerEvent** (QTimerEvent *event) Occurs when ever a timer event is called. Precondition: a QTimerEvent is called. Postcondition: If game is started and the direction has not changed, then the **Snake** is moved and collsion detection is run (checkCollision is called). Next, changedDir is set to false and QPaintEvent (repaint) is called.
void	**keyPressEvent** (QKeyEvent *event) Carries out the required actions for the specific key. Precondition: a QKeyEvent is called. (a key is pressed) Postcondition: Actions for specified key are carried out.
void	**startGame** () If game is not started, resets data members to default values, deletes **Snake**, creates a new **Snake** to default length and position, and moves food. Precondition: **Game** is not started and a 'Y' or 'N' was pressed during intro Screen 2. Postcondition: **Game** is started and **Snake** is centered on screen.
void	**pauseGame** () Pauses and Unpauses a started game. Precondition: **Game** is started and 'P' is pressed, or **Game** is paused and 'P' or a directional key is pressed. Postcondition: If game is started, the game is paused. If game is paused, game is unpaused.
void	**stopGame** () Stops the game and the timer, sets up variables to paint the gameOverScreen, and calls a paint event. Precondition: **Snake** makes an illegal collision or any key is pressed during the Blue Screen of Death. Postcondition: The paint event will display the gameOverScreen.
void	**victory** () Stops the game and the timer, sets up variables to paint the victoryScreen, and calls a paint event. Precondition: If snake collides with foodWinner, game is started and 'V' is pressed, or growCount == 315. Postcondition: The paint event will display the victoryScreen.

void	**blueScreenDeath** () Stops the timer, sets up variables to paint the Blue Screen of Death, sets points to zero, and calls a paint event. Precondition: If snake collides with specialFood, or game is started and '8' is pressed. Postcondition: Points are set to zero and the paint event will display the Blue Screen of Death.
void	**gameOverScreen** (QPainter &painter) The screen that is displayed when a **Game** Over occurs. Precondition: Paint event is called and gameOver is true. Postcondition: gameOverScreen is displayed.
void	**victoryScreen** (QPainter &painter) The screen that is displayed when a Victory occurs. Precondition: Paint event is called and gameWon is true. Postcondition: victoryScreen is displayed.
void	**getPoints** (QPainter &painter) Paints points on the status bar on the iphone. Precondition: Paint event is called and either gameStarted or BSOD is true. Postcondition: Points are displayed on status bar of iphone.
void	**introScreen** (QPainter &painter) Paints the intro screens on the screen of the iphone. Precondition: Paint event is called and either introPrompt1 or introPrompt2 is true. Postcondition: Either intro screen 1 or intro screen 2 is displayed.
void	**checkCollision** () Carries out actions if snake collided with something. Precondition: The **Snake** is moved by a directional key press or a timer event. Postcondition: Specific action carried out according to what the snake collided with.

Private Attributes

Snake *	**snake** The infinitely growing snake.
Food *	**food** The normal piece of **Food**, red or green apple logo. If the **Snake** eats this **Food**, the **Snake** will grow one **SnakeBlock**.
SpecialFood *	**specialFood** The special piece of **Food**, The Windows 8 logo. If the **Snake** eats this **Food**, the game will go into a Blue Screen of Death **Game** over in which the user loses all points. This will spawn every ten grows for 100 paint events.
FoodWinner *	**foodWinner** The winning piece of **Food**, The Linux logo. If the **Snake** eats this **Food**, the game will go into a Victory Screen in which the game ends and a special screen is displayed. This will spawn after the specialFood has been painted 100 times. It will be painted for 100 paint events, and then disappear.
Iphone	**iphone** The **Iphone** border around the game.
int	**timerId** The unique timer identifier that is returned from startTimer(). It is passed to killTimer(int id) in order to stop the timer.
int	**speedDelay** The timer delay that controls the speed. It is passed to startTimer() in order to start the timer.
unsigned int	**points** The number of points the user has accumulated.
unsigned int	**pointMulti**

	The points multiplier, this variable will be multiplied with the points that the user earns during each grow. The User sets this variable during 1st intro screens.
bool	**wrapAround** This control what logic is used when the **Snake** hits the wall. The User sets this variable during 2nd intro screens. If False, a game over will occur when the **Snake** hits the wall. If True, the **Snake** will wrap around to the other side of the screen when it hits the wall.
bool	**introPrompt1** Controls when intro prompt 1 is displayed, and what actions to take, when intro prompt 1, is displayed during a key press event.
bool	**introPrompt2** Controls when intro prompt 2 is displayed, and what actions to take, when intro prompt 2 is displayed, during a key press event.
bool	**gameStarted** Whether or not the game is started. True when game is started, otherwise False.
bool	**gameOver** Whether or not a game over has occured. True when a game over has occur, otherwise False. Controls when **Game** Over Screen is displayed.
bool	**gameWon** Whether or not a victory (the game has been won) has occured. True when a victory has occured, otherwise False. Controls when Victory Screen is displayed.
bool	**gameEnded** Whether or not a victory or a game over has occured. True when a victory or a game over has occured, otherwise False. Controls what actions to take during a key press event, when a victory or a game over has occured.
bool	**bSOD** Whether or not a Blue screen of death has occured. True when a Blue screen of death has occured, otherwise False. Controls when Blue screen of death is displayed, and what actions to take, when Blue screen of death is displayed, during a key press event.
bool	**paused** Whether or not the game is paused. True when the game is paused, otherwise False.
bool	**changedDir** Whether or not the snake has changed directions before the last tiemr event. True when the snake has changed directions before the last timer event, otherwise False.
char	**lastDir** Stores the last valid direction of the **Snake**.
unsigned int	**growCount** The number of times the snake has grown. Controls when to start displaying specialFood. Once growCount % 10 == 0, specialFood is painted.
unsigned int	**spawnWinnerCount** The number of times the special food has been painted. Controls when ro start displaying foodWinner. Once spawnWinnerCount % 1000 == 0, foodWinner is painted.
bool	**deathSpawn** Whether or not the specialFood will be painted. True when a specialFood will be painted, otherwise False. Controls when specialFood is displayed, and what actions to take, when specialFood is displayed, during a key press event or during checkCollision.

bool	**winSpawn** Whether or not the foodWinner will be painted. True when a foodWinner will be painted, otherwise False. Controls when foodWinner is displayed, and what actions to take, when foodWinner is displayed, during a key press event or during checkCollision.
int	**deathCounter** Increments everytime specialFood is painted. Controls when to stop displaying specialFood. Once deathCounter % 100 == 0, specialFood is no longer painted.
int	**winCounter** Increments everytime foodWinner is painted. Controls when to stop displaying foodWinner. Once winCounter % 100 == 0, foodWinner is no longer painted.
bool	**doOneTime** Controls whether or not to move and start displaying foodWinner or specialFood.
bool	**fixCount** Controls whether or not to fix the grow count, depending on the spawn of specialFood.
int	**switchFood** Controls which apple is to be displayed. If 0, then green apple is displayed If 1, then red apple is displayed.
QImage	**blueScreenImage** This is the QImage of the screen that will be displayed when the **Snake** eats a Windows 8 logo.
QRect	**blueScreenRect** This is the QRect of the screen that will be displayed when the **Snake** eats a Windows 8 logo.
QImage	**victoryScreenImage** This is the QImage of the screen that will be displayed when the **Snake** eats a Linux logo.
QRect	**victoryScreenRect** This is the QRect of the screen that will be displayed when the **Snake** eats a Linux logo.

Detailed Description

The **Snake Game** Logic. This class contains the logic that makes up the **Snake Game**.

Member Function Documentation

void Game::gameOverScreen (QPainter & painter) [protected]

The screen that is displayed when a **Game** Over occurs. Precondition: Paint event is called and gameOver is true. Postcondition: gameOverScreen is displayed.

Parameters:

QT Qpainter object reference.

void Game::getPoints (QPainter & painter) [protected]

Paints points on the status bar on the iphone. Precondition: Paint event is called and either gameStarted or BSOD is true. Postcondition: Points are displayed on status bar of iphone.

Parameters:

QT	Qpainter object reference.

void Game::introScreen (QPainter & painter) [protected]

Paints the intro screens on the screen of the iphone. Precondition: Paint event is called and either introPrompt1 or introPrompt2 is true. Postcondition: Either intro screen 1 or intro screen 2 is displayed.

Parameters:

QT	Qpainter object reference.

void Game::keyPressEvent (QKeyEvent * event) [protected]

Carries out the required actions for the specific key. Precondition: a QKeyEvent is called. (a key is pressed) Postcondition: Actions for specified key are carried out.

Parameters:

***event**	QT event pointer

void Game::paintEvent (QPaintEvent * event) [protected]

Paints the necessary object on the screen. Precondition: a QPaintEvent is called. Postcondition: Necessary objects are painted on the screen.

Parameters:

***event**	QT event pointer

void Game::timerEvent (QTimerEvent * event) [protected]

Occurs when ever a timer event is called. Precondition: a QTimerEvent is called. Postcondition: If game is started and the direction has not changed, then the **Snake** is moved and collsion detection is run (checkCollision is called). Next, changedDir is set to false and QPaintEvent (repaint) is called.

Parameters:

***event**	QT event pointer

void Game::victoryScreen (QPainter & painter) [protected]

The screen that is displayed when a Victory occurs. Precondition: Paint event is called and gameWon is true. Postcondition: victoryScreen is displayed.

Parameters:

QT	Qpainter object reference.

The documentation for this class was generated from the following files:

- **Game.h**
- **Game.cpp**

Snake Class Reference

The **Snake** Represents the infinitely growing snake which is made up of a vector of **SnakeBlock** pointers. More...

`#include <Snake.h>`

List of all members.

Public Member Functions

	Snake () Class Constructor - Makes a **Snake**, one block big appear. on the screen, and initializes the data members to default values Precondition: New game has started. Postcondition: **Snake** head appears in the center of the screen.
	~Snake () Class Destructor - Deletes dynamically allocated vector of **SnakeBlock** pointers.
void	**resetState** () Resets the **Snake** to one block big and in center and sets the data members to default values. Precondition: New game has started. Postcondition: **Snake** head appears in the center of the screen.
void	**goRight** (bool &wrapAround) Moves the whole **Snake** right one block and changes image to going right image, unless the **Snake** is going left and bigger than 1 **SnakeBlock**. Precondition: The user press Right arrow or the last valid direction was right. Postcondition: the whole **Snake** will be moved right one block.
void	**goLeft** (bool &wrapAround) Moves the whole **Snake** left one block and changes image to going left image, unless the **Snake** is going right and bigger than 1 **SnakeBlock**. Precondition: The user press Left arrow or the last valid direction was left. Postcondition: the whole **Snake** will be moved left one block.
void	**goUp** (bool &wrapAround) Moves the whole **Snake** up one block and changes image to going up image, unless the **Snake** is going down and bigger than 1 **SnakeBlock**. Precondition: The user press Up arrow or the last valid direction was up. Postcondition: the whole **Snake** will be moved up one block.
void	**goDown** (bool &wrapAround) Moves the whole **Snake** down one block and changes image to going down image, unless the **Snake** is going up and bigger than 1 **SnakeBlock**. Precondition: The user press Down arrow or the last valid direction was down. Postcondition: the whole **Snake** will be moved down one block.
void	**autoMove** (bool &wrapAround) Moves the **Snake** one block in the last valid direction. Precondition: Timer event occured Postcondition: The **Snake** is

	moved one block in the last valid direction.
void	**grow** () Adds one **SnakeBlock** to the end of the **Snake**. Precondition: The **Snake** head collided with a normal piece of food Postcondition: Adds one **SnakeBlock** to the end of the **Snake**.
bool	**checkBadFood** (**Food** *food) Checks if **Food** is on top of **Snake**. Precondition: A **Food** object was moved. Postcondition: Returns True, if food is on top of **Snake**.
bool	**checkCollision** () Checks if **Snake** head collided with the **Snake** body. Precondition: The **Snake** moved. Postcondition: Returns True, if the **Snake** head collided with the **Snake** body.
bool	**hitsWall** () Checks if **Snake** head collided with the wall (off the screen). Precondition: Wrap around is disabled and the **Snake** moved. Postcondition: Returns True, if the **Snake** head collided with the wall.
void	**drawSnake** (QPainter &painter) Paints the entire vector of SnakeBlocks on the screen at its current location. Precondition: Instance of **Game** called a paint event and the game is started. Postcondition: The whole **Snake** will be painted on the screen.
QRect	**getRect** () Returns the Qrect of the **Snake** head (front of snake). Precondition: Any collision detection that involes the **Snake** occured. Postcondition: The Qrect of the **Snake** head (front of snake) is returned.

Private Attributes

std::vector< **SnakeBlock** * >	**snakeBlocks** Vector of **SnakeBlock** pointers.
int	**x** Current or future x position of front block (head of snake).
int	**y** Current or future y position of front block (head of snake).
int	**xDir** Current x direction of **Snake**.
int	**yDir** Current y direction of the **Snake**.
int	**growsCount** Number of times the snake has grown.
int	**switchBody** Used to switch between **Snake** body block images. When 0, **SnakeBlock** will be blue. When 1, **SnakeBlock** will be purple.

Detailed Description

The **Snake** Represents the infinitely growing snake which is made up of a vector of **SnakeBlock** pointers.

Member Function Documentation

void Snake::autoMove (bool & wrapAround)

Moves the **Snake** one block in the last valid direction. Precondition: Timer event occured Postcondition: The **Snake** is moved one block in the last valid direction.

Parameters:

&wrapAround If true, allows wrap around.

bool Snake::checkBadFood (Food * food)

Checks if **Food** is on top of **Snake**. Precondition: A **Food** object was moved. Postcondition: Returns True, if food is on top of **Snake**.

Parameters:

***food** **Food** object pointer. It will be used to check if it is on top of the **Snake**.

Returns:

Whether the **Food** is on top of the **Snake**.

bool Snake::checkCollision ( )

Checks if **Snake** head collided with the **Snake** body. Precondition: The **Snake** moved. Postcondition: Returns True, if the **Snake** head collided with the **Snake** body.

Returns:

Whether the **Snake** head collided with the **Snake** body.

void Snake::drawSnake (QPainter & painter)

Paints the entire vector of SnakeBlocks on the screen at its current location. Precondition: Instance of **Game** called a paint event and the game is started. Postcondition: The whole **Snake** will be painted on the screen.

Parameters:

&painter QPainter object created in **Game** to paint images on screen.

QRect Snake::getRect ( )

Returns the Qrect of the **Snake** head (front of snake). Precondition: Any collision detection that involes the **Snake** occured. Postcondition: The Qrect of the **Snake** head (front of snake) is returned.

Returns:

Returns Qrect of the **Snake** head (front of snake).

void Snake::goDown (bool & wrapAround)

Moves the whole **Snake** down one block and changes image to going down image, unless the **Snake** is going up and bigger than 1 **SnakeBlock**. Precondition: The user press Down arrow or the last valid direction was down. Postcondition: the whole **Snake** will be moved down one block.

Parameters:

&wrapAround If true, allows wrap around.

void Snake::goLeft (bool & wrapAround)

Moves the whole **Snake** left one block and changes image to going left image, unless the **Snake** is going right and bigger than 1 **SnakeBlock**. Precondition: The user press Left arrow or the last valid direction was left. Postcondition: the whole **Snake** will be moved left one block.

Parameters:

&wrapAround If true, allows wrap around.

void Snake::goRight (bool & wrapAround)

Moves the whole **Snake** right one block and changes image to going right image, unless the **Snake** is going left and bigger than 1 **SnakeBlock**. Precondition: The user press Right arrow or the last valid direction was right. Postcondition: the whole **Snake** will be moved right one block.

Parameters:

&wrapAround If true, allows wrap around.

void Snake::goUp (bool & wrapAround)

Moves the whole **Snake** up one block and changes image to going up image, unless the **Snake** is going down and bigger than 1 **SnakeBlock**. Precondition: The user press Up arrow or the last valid direction was up. Postcondition: the whole **Snake** will be moved up one block.

Parameters:

&wrapAround If true, allows wrap around.

bool Snake::hitsWall ( )

Checks if **Snake** head collided with the wall (off the screen). Precondition: Wrap around is disabled and the **Snake** moved. Postcondition: Returns True, if the **Snake** head collided with the wall.

Returns:

Whether the **Snake** head collided the wall.

The documentation for this class was generated from the following files:

- **Snake.h**

- **Snake.cpp**

SnakeBlock Class Reference

The **SnakeBlock** Represents one block of the **Snake**. More...

`#include <SnakeBlock.h>`

List of all members.

Public Member Functions

	SnakeBlock (int xCoord, int yCoord, int imgSelect) Class Constructor - Creates one **SnakeBlock** at the given postion with the specified image and initializes the data members to default values. Precondition: A new game was created or the **Snake** is growing bigger. Postcondition: A new Snakeblock is created and added to the **Snake**.
	~SnakeBlock () Class Destructor - Does Nothing.
void	**setPosition** (int **x**, int **y**) Moves the **SnakeBlock** to the given position. Precondition: A **SnakeBlock** object needed to be moved or placed on the screen. Postcondition: The **SnakeBlock** will be moved to the given position.
int	**getxPos** () Returns the Current x position of the **SnakeBlock**. Precondition: **Snake** object requested Current x position of **SnakeBlock**. Postcondition: Current x position is returned.
int	**getyPos** () Returns the Current y position of the **SnakeBlock**. Precondition: **Snake** object requested Current y position of **SnakeBlock**. Postcondition: Current y position is returned.
int	**getxOldPos** () Returns the Previous x position of the **SnakeBlock**. Precondition: **Snake** object requested Previous x position of **SnakeBlock**. Postcondition: Previous x position is returned.
int	**getyOldPos** () Returns the Previous y position of the **SnakeBlock**. Precondition: **Snake** object requested Previous y position of **SnakeBlock**. Postcondition: Previous y position is returned.
void	**autoMove** (int pxCoord, int pyCoord) Moves the **SnakeBlock** to the location the **SnakeBlock** in front of it. Precondition: **SnakeBlock** in front of this **SnakeBlock** was moved. Postcondition: This **SnakeBlock** has been moved to the location the **SnakeBlock** in front of it.
bool	**hitsWall** () Checks if **SnakeBlock** is off the screen (collided with the wall). Precondition: checkCollision function in game was called and wrap around is disabled. Postcondition: If True, gameOver will occur.
void	**setImageDown** () Switches the image of the **SnakeBlock** (**Snake** head only) to the going down image. Precondition: **Snake** changed direction to going down. Postcondition: The image of snakeblockdown.png is displayed.
void	**setImageUp** ()

	Switches the image of the **SnakeBlock** (**Snake** head only) to the going up image. Precondition: **Snake** changed direction to going up. Postcondition: The image of snakeblockup.png is displayed.
void	**setImageLeft** () Switches the image of the **SnakeBlock** (**Snake** head only) to the going left image. Precondition: **Snake** changed direction to going left. Postcondition: The image of snakeblockleft.png is displayed.
void	**setImageRight** () Switches the image of the **SnakeBlock** (**Snake** head only) to the going right image. Precondition: **Snake** changed direction to going right. Postcondition: The image of snakeblockright.png is displayed.
QRect	**getRect** () Returns the rect of **SnakeBlock**. Precondition: Any collision detection or paint event that involes the **SnakeBlock** occured. Postcondition: The Qrect of the **SnakeBlock** is returned.
QImage &	**getImage** () Returns the image of the **SnakeBlock**. Precondition: Instance of **Game** called a paint event and the game is started. Postcondition: The image of the **SnakeBlock** will be painted on the screen.

Private Attributes

int	**x** Current x position of the **SnakeBlock**.
int	**y** Current y position of the **SnakeBlock**.
int	**px** Previous x position of the **SnakeBlock**.
int	**py** Previous y position of the **SnakeBlock**.
QImage	**image** The QImage of **SnakeBlock**.
QRect	**rect** The QRect of **SnakeBlock**.

Detailed Description

The **SnakeBlock** Represents one block of the **Snake**.

Constructor & Destructor Documentation

```
SnakeBlock::SnakeBlock ( int  xCoord,
                         int  yCoord,
                         int  imgSelect
                       )
```

Class Constructor - Creates one **SnakeBlock** at the given postion with the specified image and initializes the data members to default values. Precondition: A new game was created or the **Snake** is growing bigger. Postcondition: A new Snakeblock is created and added to the **Snake**.

Parameters:

xCoord	x position to place **SnakeBlock**.
yCoord	y position to place **SnakeBlock**.
imgSelect	Chooses between the different **SnakeBlock** images.

Member Function Documentation

```
void SnakeBlock::autoMove ( int  pxCoord,
                            int  pyCoord
                          )
```

Moves the **SnakeBlock** to the location the **SnakeBlock** in front of it. Precondition: **SnakeBlock** in front of this **SnakeBlock** was moved. Postcondition: This **SnakeBlock** has been moved to the location the **SnakeBlock** in front of it.

Parameters:

pxCoord	The previous x position of the **SnakeBlock** in front of it.
pyCoord	The previous y position of the **SnakeBlock** in front of it.

```
QImage & SnakeBlock::getImage (  )
```

Returns the image of the **SnakeBlock**. Precondition: Instance of **Game** called a paint event and the game is started. Postcondition: The image of the **SnakeBlock** will be painted on the screen.

Returns:

QImage The image of the **SnakeBlock**.

```
QRect SnakeBlock::getRect (  )
```

Returns the rect of **SnakeBlock**. Precondition: Any collision detection or paint event that involes the **SnakeBlock** occured. Postcondition: The Qrect of the **SnakeBlock** is returned.

Returns:

The rect of **SnakeBlock**.

int SnakeBlock::getxOldPos ( )

Returns the Previous x position of the **SnakeBlock**. Precondition: **Snake** object requested Previous x position of **SnakeBlock**. Postcondition: Previous x position is returned.

Returns:
The Previous x position of the **SnakeBlock**.

int SnakeBlock::getxPos ( )

Returns the Current x position of the **SnakeBlock**. Precondition: **Snake** object requested Current x position of **SnakeBlock**. Postcondition: Current x position is returned.

Returns:
The Current x position of the **SnakeBlock**.

int SnakeBlock::getyOldPos ( )

Returns the Previous y position of the **SnakeBlock**. Precondition: **Snake** object requested Previous y position of **SnakeBlock**. Postcondition: Previous y position is returned.

Returns:
The Previous y position of the **SnakeBlock**.

int SnakeBlock::getyPos ( )

Returns the Current y position of the **SnakeBlock**. Precondition: **Snake** object requested Current y position of **SnakeBlock**. Postcondition: Current y position is returned.

Returns:
The Current y position of the **SnakeBlock**.

bool SnakeBlock::hitsWall ( )

Checks if **SnakeBlock** is off the screen (collided with the wall). Precondition: checkCollision function in game was called and wrap around is disabled. Postcondition: If True, gameOver will occur.

Returns:
Whether the SnakeBlcok is off the screen (collided with the wall).

```
void SnakeBlock::setPosition ( int  xCoord,
                               int  yCoord
                             )
```

Moves the **SnakeBlock** to the given position. Precondition: A **SnakeBlock** object needed to be moved or placed on the screen. Postcondition: The **SnakeBlock** will be moved to the given position.

Parameters:

x	The x position to move the **SnakeBlock** to.
y	The y position to move the **SnakeBlock** to.

The documentation for this class was generated from the following files:

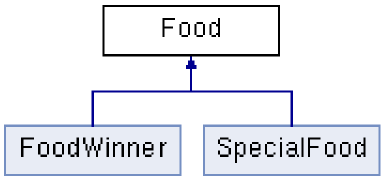
- **SnakeBlock.h**
- **SnakeBlock.cpp**

Food Class Reference

A **Food** piece The parent of all food pieces. If not inherited, **Food** represents a red or green Apple Logo. More...

`#include <Food.h>`

Inheritance diagram for Food:



List of all members.

Public Member Functions

	Food () Class Constructor - loads first image, moves to random location. Precondition: A new instance of game was created. Postcondition: A new piece of food is created with default image.
	~Food () Class Destructor - Does Nothing.
void	**moveFood** (**Food** *otherFood) Draws **Food** in a random location. Precondition: The **Snake** head collided with a piece of **Food** or **Food** cheat code key was pressed. Postcondition: **Food** is moved to a random location on the screen which is not on top of otherFood.
void	**moveFood** (**Food** *otherFood, int whichFood) Draws **Food** in a random location. Precondition: The **Snake** head collided with a piece of **Food** or **Food** cheat code key was pressed. Postcondition: **Food** is moved to a random location on the screen which is not on top of otherFood.
void	**moveFood** (int whichFood) Draws **Food** in a random location. Precondition: The **Snake** head collided with a piece of **Food** or **Food** cheat code key was pressed. Postcondition: **Food** is moved to a random location on the screen which is not on top of otherFood.
void	**pickLocation** () Picks a random location to move **Food** to, but does not move **Food**. Precondition: **Food** needs to be moved to a new location. Postcondition: New location is picked for **Food** to move to.
QRect	**getRect** () Returns rect of **Food**. Precondition: Any collision detection or paint event that involes the **Food** occured. Postcondition: The Qrect of the **Food** is returned.
QImage &	**getImage** () Returns image of **Food**. Precondition: Instance of **Game** called a paint event and the game is started. Postcondition: The image of the **Food** will be painted on the screen.

Protected Attributes

int	**x** The x position of **Food**.

int	**y** The y position of **Food**.
QImage	**image** The QImage of **Food**.
QRect	**rect** The QRect of **Food**.

Detailed Description

A **Food** piece The parent of all food pieces. If not inherited, **Food** represents a red or green Apple Logo.

Member Function Documentation

QImage & Food::getImage ( )

Returns image of **Food**. Precondition: Instance of **Game** called a paint event and the game is started. Postcondition: The image of the **Food** will be painted on the screen.

Returns:
Image of **Food**.

QRect Food::getRect ( )

Returns rect of **Food**. Precondition: Any collision detection or paint event that involes the **Food** occured. Postcondition: The Qrect of the **Food** is returned.

Returns:
The rect of **Food**.

void Food::moveFood (Food * otherFood)

Draws **Food** in a random location. Precondition: The **Snake** head collided with a piece of **Food** or **Food** cheat code key was pressed. Postcondition: **Food** is moved to a random location on the screen which is not on top of otherFood.

Parameters:

***otherFood**	The other piece of **Food** on the screen
***otherFood**	The other piece of **Food** on the screen.

```
void Food::moveFood ( Food *  otherFood,
                      int     whichFood
                    )
```

Draws **Food** in a random location. Precondition: The **Snake** head collided with a piece of **Food** or **Food** cheat code key was pressed. Postcondition: **Food** is moved to a random location on the screen which is not on top of otherFood.

Parameters:

***otherFood**	The other piece of **Food** on the screen.
whichFood	The integer switch between **Food** images.

```
void Food::moveFood ( int  whichFood )
```

Draws **Food** in a random location. Precondition: The **Snake** head collided with a piece of **Food** or **Food** cheat code key was pressed. Postcondition: **Food** is moved to a random location on the screen which is not on top of otherFood.

Parameters:

whichFood	The integer switch between **Food** images.

The documentation for this class was generated from the following files:

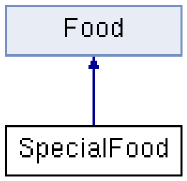
- **Food.h**
- **Food.cpp**

SpecialFood Class Reference

A **SpecialFood** piece A daughter of the **Food** class. Represents a Windows 8 Logo. If **Snake** collides with **SpecialFood**, a special game over will occur (BSOD). More...

```
#include <SpecialFood.h>
```

Inheritance diagram for SpecialFood:



List of all members.

Public Member Functions

SpecialFood ()
~SpecialFood ()

Detailed Description

A **SpecialFood** piece A daughter of the **Food** class. Represents a Windows 8 Logo. If **Snake** collides with **SpecialFood**, a special game over will occur (BSOD).

Constructor & Destructor Documentation

SpecialFood::SpecialFood ( )

Class Constructor - loads first image, moves to random location. Precondition: A new instance of game was created. Postcondition: A new piece of food is created with Windows 8 logo image.

SpecialFood::~SpecialFood ( )

Class Destructor - Does Nothing

The documentation for this class was generated from the following files:

- **SpecialFood.h**
- **SpecialFood.cpp**

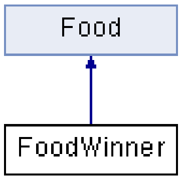
Generated on Sat Oct 13 2012 23:03:45 for Homework 2 Design Document by doxygen 1.7.6.1

FoodWinner Class Reference

A **FoodWinner** piece A daughter of the **Food** class. Represents a Linux Logo. If **Snake** collides with **FoodWinner**, a victory screen will occur. More...

```
#include <FoodWinner.h>
```

Inheritance diagram for FoodWinner:



List of all members.

Public Member Functions

FoodWinner ()
~FoodWinner ()

Detailed Description

A **FoodWinner** piece A daughter of the **Food** class. Represents a Linux Logo. If **Snake** collides with **FoodWinner**, a victory screen will occur.

Constructor & Destructor Documentation

FoodWinner::FoodWinner ( )

Class Constructor - loads first image, moves to random location. Precondition: A new instance of game was created. Postcondition: A new piece of food is created with Linux logo image.

FoodWinner::~FoodWinner ( )

Class Destructor - Does Nothing

The documentation for this class was generated from the following files:

- **FoodWinner.h**
- **FoodWinner.cpp**

Generated on Sat Oct 13 2012 23:03:45 for Homework 2 Design Document by doxygen 1.7.6.1

Iphone Class Reference

A **Iphone** image and location of where to paint. More...

```
#include <iphone.h>
```

List of all members.

Public Member Functions

	Iphone () Class Constructor - loads image, moves to specified location. Precondition: A new instance of game was created. Postcondition: A new **Iphone** image is loaded and ready to be painted.
	~Iphone () Class Destructor - Does Nothing.
void	**setPosition** () Sets the location of the **Iphone**. Precondition: A **Iphone** object was created. Postcondition: Location is set for the **Iphone**.
QRect	**getRect** () Returns rect of **Iphone**. Precondition: Instance of **Game** called a paint event. Postcondition: The Qrect of the **Iphone** is returned.
QImage &	**getImage** () Returns image of **Iphone**. Precondition: Instance of **Game** called a paint event. Postcondition: The image of the **Iphone** will be painted on the screen.

Protected Attributes

QImage	**image** The QImage of **Iphone**.
QRect	**rect** The QRect of **Iphone**.

Detailed Description

A **Iphone** image and location of where to paint.

Member Function Documentation

QImage & Iphone::getImage ( )

Returns image of **Iphone**. Precondition: Instance of **Game** called a paint event. Postcondition: The image of the **Iphone** will be painted on the screen.

Returns:

Image of **Iphone**.

QRect Iphone::getRect ( )

Returns rect of **Iphone**. Precondition: Instance of **Game** called a paint event. Postcondition: The Qrect of the **Iphone** is returned.

Returns:

The rect of **Iphone**.

The documentation for this class was generated from the following files:

- **iphone.h**
- **iphone.cpp**

Global Data/Functions

- Global Functions
    - void center (QWidget &widget)
        - Sets up window size and centers window on screen.
        - Precondition: Program has started.
        - Postcondition: **Game** window is displayed on center of screen.

High-level Architecture

When the program starts, the QT window is initialized and the basic window settings are set such as the size of the window. Then Game is launched. Next, the required objects are created from the following classes: Snake, Food, SpecialFood, WinningFood, and iPhone. When an object of the Snake class is created, a vector of SnakeBlock objects is created. After that, the paint event is called which displays intro screen 1. The user is asked to select the speed which will set the timer delay and the points multiplier. Once the user makes a selection, another paint event is called which displays intro screen 2. During intro screen 2, the user is asked to whether to enable or disable wrap around which will set wraparound to TRUE or FALSE. Once the user makes a selection, startGame is called which starts the timer, resets data members to default values, deletes Snake, creates a new Snake dynamically, and moves food using moveFood and checkBadFood methods of the Food class. Then, another paint event is called which displays the game play screen.

Now, the game begins. Once the user presses an arrow key, the snake begins to move by the autoMove method of the Snake class and collision detection occurs after each time the snake moves by the checkCollision method of the game class. If during checkCollision the snake makes an illegal collision, then the gameOver function is called and the game either restarts or the program terminates. If during checkCollision the snake collides with an Apple logo, the snake grows 1 SnakeBlock using snake->grow and points are added to points. If during checkCollision the snake collides with a Windows 8 logo, the blueScreenDeath function is called which displays the bsod. If during checkCollision the snake collides with a Linux logo, the victory screen is displayed using the victory function. If the user presses ‘P’, pauseGame function is called and game is paused.

If the user presses ESC at any time, qApp->exit function is called and program is terminated.

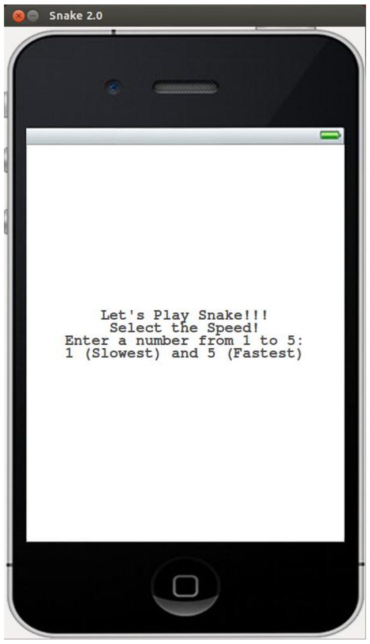
If the user presses ‘R’ at any time, stopGame function is called, introPrompt1 is set to true, and intro screen 1 is displayed.

User Interface

Universal Controls

- Esc key is pressed
    - Exits app
- 'R' key is pressed
    - **Game** is reset
    - Intro Screen 1 is displayed

Intro Screen 1



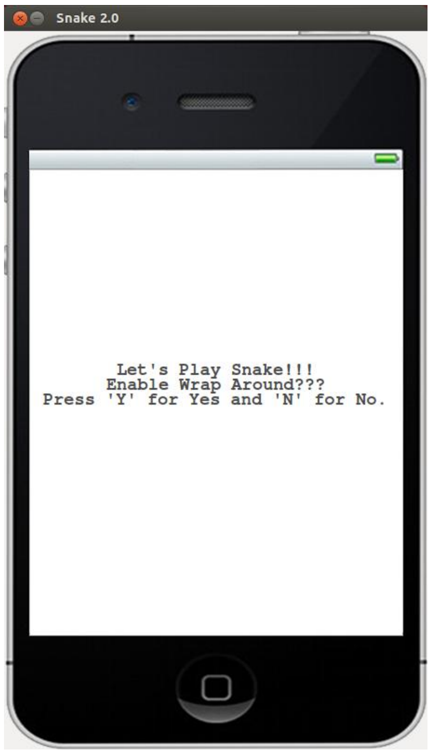
Let's Play Snake!!!
Select the Speed!
Enter a number from 1 to 5:
1 (Slowest) and 5 (Fastest)

Intro Screen 1 Controls

- '1' key is pressed
    - pointMulti = 1;
    - speedDelay = 200;
    - Intro Screen 2 is displayed
- '2' key is pressed
    - pointMulti = 2;
    - speedDelay = 169;
    - Intro Screen 2 is displayed
- '3' key is pressed
    - pointMulti = 3;
    - speedDelay = 138;
    - Intro Screen 2 is displayed
- '4' key is pressed
    - pointMulti = 4;
    - speedDelay = 107;
    - Intro Screen 2 is displayed
- '5' key is pressed
    - pointMulti = 5;
    - speedDelay = 75;
    - Intro Screen 2 is displayed
- '9' key is pressed
    - pointMulti = 1000;

- speedDelay = 0;
- Intro Screen 2 is displayed

Intro Screen 2

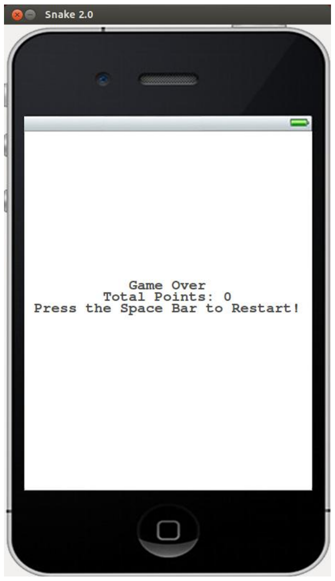


```
Let's Play Snake!!!
Enable Wrap Around???
Press 'Y' for Yes and 'N' for No.
```

Intro Screen 2 Controls

- 'Y' key is pressed
  - wrapAround = TRUE;
  - **Game** is started
- 'N' key is pressed
  - wrapAround = FALSE;
  - **Game** is started

Game Over Screen

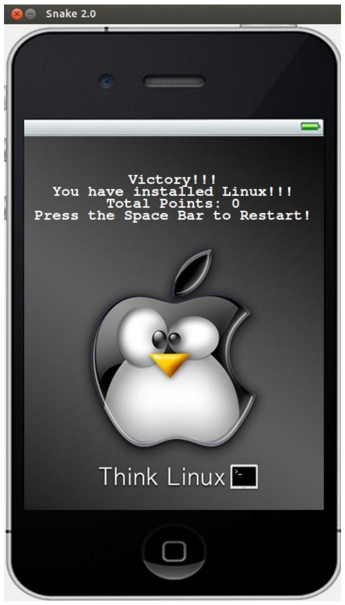


```
Game Over
Total Points: 0
Press the Space Bar to Restart!
```

Game Over Screen Controls

- Space bar
    - Intro Screen 1 is displayed

Victory Screen

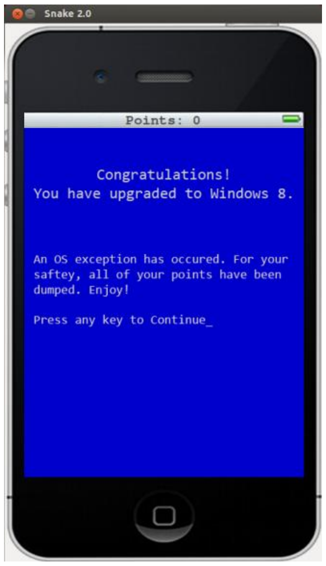


```
Victory!!!
You have installed Linux!!!
Total Points: 0
Press the Space Bar to Restart!
```

Victory Screen Controls

- Space bar is pressed
    - Intro Screen 1 is displayed

Blue Screen of Death



Congratulations!
You have upgraded to Windows 8.

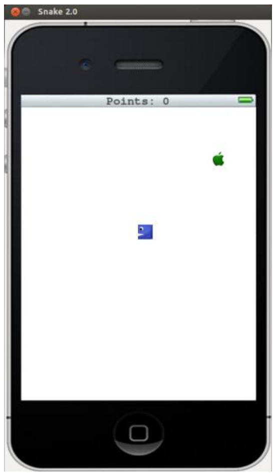
An OS exception has occured. For your saftey, all of your points have been dumped. Enjoy!

Press any key to Continue_

Blue Screen of Death Controls

- Any key is pressed, expect 'R' or ESC
    - **Game** Over Screen is displayed

Game Play Screen



Game Play Controls

- Normal **Game** Play Controls
    - Right arrow key is pressed
        - **Snake** moves right
    - Left arrow key is pressed
        - **Snake** moves left
    - Up arrow key is pressed
        - **Snake** moves up
    - Down arrow key is pressed
        - **Snake** moves down

    - 'P' key is pressed
        - Pauses game (**Snake** stops moving)
- Paused **Game** Play Controls
    - Right arrow key is pressed
        - **Game** resumes
        - **Snake** moves right
    - Left arrow key is pressed
        - **Game** resumes
        - **Snake** moves left
    - Up arrow key is pressed
        - **Game** resumes
        - **Snake** moves up
    - Down arrow key is pressed
        - **Game** resumes
        - **Snake** moves down
    - 'P' key is pressed
        - **Game** resumes
        - **Snake** moves in the last direction
- Cheat Codes (Only active during Normal **Game** Play)
    - 'M' key or 'A' key is pressed
        - switches food images
        - Moves the apple
    - 'G' key is pressed
        - Grows snake, but doesn't increase points
    - 'W' key is pressed
        - Moves specialFood (the Windows 8 logo)
        - Paints specialFood
    - '8' key is pressed
        - Displays blue screen of death
        - Causes BSOD **Game** Over
    - 'L' key is pressed
        - Moves foodWinner (the Linux logo)
        - Paints foodWinner
    - 'V' key is pressed
        - Displays Victory screen
        - Causes Victory