My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Flappy_Bird

Functions

- void setupGameInfo ()

  SetupGameInfo, sets up game based on difficulty.
- void setDifficulty (int choice)

  Sets game difficulty.
- void updateScores ()

  Updates the scoreboard.
- void createPipeQueue ()

  Assigns a pipe queue to the game state.
- bool hits (Pipe *p)

  Detects collisions.
- void initPipes ()

  Initialises a pipe.
- void updateAllPipes ()

  Updates all pipes in the queue.
- bool isbirdAlive ()

  If the bird is alive or not.
- void initBird ()

  Initialises a bird.
- void upBirdy (void)

  Moves the bird up.
- void updateBirdy (void)

  Updates the birds state.
- void initCoin ()

  Initialises a coin.
- void updateCoin ()

  Updates coin positioning.
- void drawProgress ()

  Draws progress circles.
- void drawBirdChoice ()

  Draws the bird selector.
- void drawHighscores (void *pData)

Displays scores and difficulty associated with that score.

- void displayLeaderboard (TOUCH_STATE tsc_state, GUI_RECT Rect)

  Creates a window to display scores and select difficulty.
- void initPregame (TOUCH_STATE tsc_state, GUI_RECT Rect)

  Initilise pregame.
- void drawEverything (GUI_RECT Rect)

  Allocates memory for drawing elements.

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmCowboyBirdy**
- GUI_CONST_STORAGE GUI_BITMAP **bmBlueBird**
- GUI_CONST_STORAGE GUI_BITMAP **bmKingBirdy**
- GUI_CONST_STORAGE GUI_BITMAP **bmBackground**
- GUI_CONST_STORAGE GUI_BITMAP **bmCoin**
- GUI_CONST_STORAGE GUI_BITMAP **bmPregame**
- GUI_CONST_STORAGE GUI_BITMAP **bmEasy**
- GUI_CONST_STORAGE GUI_BITMAP **bmHard**
- GUI_CONST_STORAGE GUI_BITMAP **bmAbsurd**
- GUI_CONST_STORAGE GUI_BITMAP **bmArrow**
- GUI_CONST_STORAGE GUI_BITMAP **bmlogo2**
- GUI_CONST_STORAGE GUI_BITMAP **bmEgg**
- GameInfo **gameInfo**
- int **turn** = 0
- bool **xpSet** = false

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 createPipeQueue()

```
void createPipeQueue (
             void  )
```

Assigns a pipe queue to the game state.

Parameters

None	

Return values

None	

queueCreate methods is called and returned value is saved within gameinfo structs queue

4.1.2.2 displayLeaderboard()

```
void displayLeaderboard (
             TOUCH_STATE tsc_state,
             GUI_RECT Rect )
```

Creates a window to display scores and select difficulty.

Parameters

TOUCH_STATE	where the screen has been pressed, GUI_RECT where to display

Return values

None.	

Sets the game mode based on what button is pressed and calls **drawHighScores** to display scores and buttons.

4.1.2.3 drawBirdChoice()

```
void drawBirdChoice ( )
```

Draws the bird selector.

Parameters

None	

Return values

None.	

Note

lv 0 = normal, lv3 = cowboy bird, lv5 = king bird

Gives an option on what bird the game is to be played with

4.1.2.4 drawEverything()

```
void drawEverything (
             GUI_RECT Rect )
```

Allocates memory for drawing elements.

Parameters

None	

Return values

None.	

Allocates the memory for **drawGame** as well as incrementing games frame-counter

4.1.2.5 drawHighscores()

```
void drawHighscores (
            void * pData )
```

Displays scores and difficulty associated with that score.

Parameters

None	

Return values

None	

Note

Normal = Green, Hard = Orange, Insane = Red

Method that displays all scores saved within the game state, displaying newest score first in the colour of difficulty played. Also displays the game states highest score achieved.

4.1.2.6 drawProgress()

```
void drawProgress ( )
```

Draws progress circles.

Parameters

None	

Return values

None	

Draws progress circles that grow the closer the user is to obtaining the next level

4.1.2.7 hits()

```
bool hits (
            Pipe * p )
```

Detects collisions.

Parameters

pointer	to the head pipe

Return values

bool	True of False

Takes in the head of the pipe queue to get the position on screen, if the birds position is in the same co-ordinates true is returned otherwise false.

4.1.2.8 initBird()

```
void initBird (
             void  )
```

Initialises a bird.

Parameters

None	

Return values

None	

Initialises a bird and adds it to the game states

4.1.2.9 initCoin()

```
void initCoin (
             void  )
```

Initialises a coin.

Parameters

None	

Return values

None	

Initialises a coin to a random position in the game state

4.1.2.10 initPipes()

```
void initPipes (
            void  )
```

Initialises a pipe.

Parameters

None	

Return values

None	

Creates a pipe and enques pipe to the game info pipe queue

4.1.2.11 initPregame()

```
void initPregame (
            TOUCH_STATE tsc_state,
            GUI_RECT Rect )
```

Initilise pregame.

Parameters

TOUCH_STATE	where the screen has been pressed, GUI_RECT where to display

Return values

None.	

Creates the window for the pregame to be drawn and waits for input

4.1.2.12 isbirdAlive()

```
bool isbirdAlive (
            void  )
```

If the bird is alive or not.

Parameters

None	

Return values

bool	true or false

Returns the gameinfo struct alive boolean variable that is altered by **updateScores**

4.1.2.13 setDifficulty()

```
void setDifficulty (
            int choice )
```

Sets game difficulty.

Parameters

int	difficulty choice, num between 0-2

Return values

None	

Note

0 = Normal, 1 = Hard, 2 = Insane

sets gameInfo struct game difficulty to later be used by **setupGameInfo**

4.1.2.14 setupGameInfo()

```
void setupGameInfo (
            void  )
```

SetupGameInfo, sets up game based on difficulty.

Parameters

None	

Return values

None	

Based on gameinfo struct difficulty variable this method sets up the speed of the game, distance between concurrent pipes and the gap between the middle of a pipe.

4.1.2.15 upBirdy()

```
void upBirdy (
             void  )
```

Moves the bird up.

Parameters

None	

Return values

None	

Alters the bird position based on gravity, moving it up

4.1.2.16 updateAllPipes()

```
void updateAllPipes (
             void  )
```

Updates all pipes in the queue.

Parameters

None	

Return values

None	

Note

also changes pipe gap size and position for insane dif

udpates co-ordinates for all pipes currently in the pipequeue, as well as calling **hits** to see if bird is still alive

4.1.2.17 updateBirdy()

```
void updateBirdy (
             void  )
```

Updates the birds state.

Parameters

None	

Return values

None	

Updates the birds y positioning (up and down movements), as well as checking the bird hasn't fell below the screen line calling **updatesScores** and ending a game loop

4.1.2.18 updateCoin()

```
void updateCoin (
             void  )
```

Updates coin positioning.

Parameters

None	

Return values

None	

Update coins x position to move it across the screen, calling **initCoin** when a new coin needs initalising (every 1170 frames)

4.1.2.19 updateScores()

```
void updateScores ( )
```

Updates the scoreboard.

Parameters

None	

Return values

None	

Once a bird dies, this method updates the scoreboard in order of most recent 1st and what difficulty the score was achieved on

4.1.3 Variable Documentation

4.1.3.1 bmHard

```
GUI_CONST_STORAGE GUI_BITMAP bmHard
```

Initial value:

```
= {
  80,
  50,
  320,
  32,
  (unsigned char *)_acHard,
  NULL,
  GUI_DRAW_BMP8888
}
```

4.1.3.2 bmKingBirdy

```
GUI_CONST_STORAGE GUI_BITMAP bmKingBirdy
```

Initial value:

```
= {
  34,
  31,
  136,
  32,
  (unsigned char *)_acKingBirdy,
  NULL,
  GUI_DRAW_BMP8888
}
```

4.1.3.3 bmlogo2

```
GUI_CONST_STORAGE GUI_BITMAP bmlogo2
```

Initial value:

```
= {
  240,
  52,
  960,
  32,
  (unsigned char *)_aclogo2,
  NULL,
  GUI_DRAW_BMP8888
}
```

4.1.3.4 bmPregame

```
GUI_CONST_STORAGE GUI_BITMAP bmPregame
```

Initial value:

```
= {
  120,
  141,
  480,
  32,
  (unsigned char *)_acPregame,
  NULL,
  GUI_DRAW_BMP8888
}
```

4.2 Game_Start_Functions

Functions

- void initGame ()
  Initialise the game state.
- void MainTask (void)
  Task container.
- int main (void)
  Main.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 initGame()

```
void initGame (
              void  )
```

Initialise the game state.

Parameters

None	

Return values

None	

Calling **initGame()** creates the initial game state by creating a bird, setting all game variables, creating the pipe queue and adding an extra points coin in.

4.2.2.2 main()

```
int main (
              void  )
```

Main.

Parameters

None	

Return values

int	

Configures all hardware used and calls **MainTask**

4.2.2.3 MainTask()

```
void MainTask (
            void  )
```

Task container.

Parameters

None	

Return values

None.	

Holds all method calls to run the game, a while loop to setup the game and an inner loop to update all positions of elements on screen whilst the bird is alive

4.3 Queue_Structure

Functions

- queue ∗ queueCreate (void)
  Creates empty queue structure.
- void enq (queue ∗q, Pipe ∗p)
  Push pipe to queue.
- int queueEmpty (const struct queue ∗q)
  Checks if the queue is empty.
- bool isOffScreen (queue ∗q)
  Checks if the queue is off screen.
- void deq (queue ∗q)
  Removes pipe at the head of the queue.
- void queueDestroy (struct queue ∗q)
  Removes entire queue and frees memory.
- void updatePipes (queue ∗q)
  Updates pipe positions.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 deq()

```
void deq (
            queue * q )
```

Removes pipe at the head of the queue.

Parameters

queue	∗q

Return values

None	

4.3.2.2 enq()

```
void enq (
            queue * q,
            Pipe * p )
```

Push pipe to queue.

Parameters

queue	*q, Pipe *p

Return values

None	

Method pushes given pipe to the top of a given queue

4.3.2.3 isOffScreen()

```
bool isOffScreen (
            queue * q )
```

Checks if the queue is off screen.

Parameters

queue	*q

Return values

bool	

4.3.2.4 queueCreate()

```
queue* queueCreate (
            void  )
```

Creates empty queue structure.

Parameters

None	

Return values

queue	

4.3.2.5 queueDestroy()

```
void queueDestroy (
            struct queue * q )
```

Removes entire queue and frees memory.

Parameters

queue	*q

Return values

None	

4.3.2.6 queueEmpty()

```
int queueEmpty (
            const struct queue * q )
```

Checks if the queue is empty.

Parameters

None	

Return values

None	

4.3.2.7 updatePipes()

```
void updatePipes (
            queue * q )
```

Updates pipe positions.

Parameters

queue	*q

Return values

None	

Chapter 5

Class Documentation

5.1 Bird Struct Reference

Models a bird.

`#include <main.h>`

Public Attributes

- int x
- int y
- int gravity
- int velocity
- int lift
- bool up

5.1.1 Detailed Description

Models a bird.

Structure to model all information needed to draw a bird.

5.1.2 Member Data Documentation

5.1.2.1 gravity

`int Bird::gravity`

Strength of the gravity

5.1.2.2 lift

```
int Bird::lift
```

?

5.1.2.3 up

```
bool Bird::up
```

Weather the bird is moving up or down

5.1.2.4 velocity

```
int Bird::velocity
```

Velocity speed

5.1.2.5 x

```
int Bird::x
```

x co-ordinate of birds position

5.1.2.6 y

```
int Bird::y
```

y co-ordinate of birds position

The documentation for this struct was generated from the following file:

- main.h

5.2 Coin Struct Reference

Models a coin.

```
#include <main.h>
```

Public Attributes

- int x
- int y

5.2.1 Detailed Description

Models a coin.

Structure to model all information needed to draw a coin.

5.2.2 Member Data Documentation

5.2.2.1 x

```
int Coin::x
```

x co-ordinate of coins position

5.2.2.2 y

```
int Coin::y
```

y co-ordinate of coins position

The documentation for this struct was generated from the following file:

- main.h

5.3 DIR Struct Reference

Public Attributes

- FATFS * **fs**
- WORD **id**
- WORD **index**
- DWORD **sclust**
- DWORD **clust**
- DWORD **sect**
- BYTE * **dir**
- BYTE * **fn**
- UINT **lockid**
- WCHAR * **lfn**
- WORD **lfn_idx**
- const TCHAR * **pat**

The documentation for this struct was generated from the following file:

- ff.h

5.4 FATFS Struct Reference

Public Attributes

- BYTE **fs_type**
- BYTE **drv**
- BYTE **csize**
- BYTE **n_fats**
- BYTE **wflag**
- BYTE **fsi_flag**
- WORD **id**
- WORD **n_rootdir**
- DWORD **last_clust**
- DWORD **free_clust**
- DWORD **cdir**
- DWORD **n_fatent**
- DWORD **fsize**
- DWORD **volbase**
- DWORD **fatbase**
- DWORD **dirbase**
- DWORD **database**
- DWORD **winsect**
- BYTE **win** [_MAX_SS]

The documentation for this struct was generated from the following file:

- ff.h

5.5 FIL Struct Reference

Public Attributes

- FATFS * **fs**
- WORD **id**
- BYTE **flag**
- BYTE **err**
- DWORD **fptr**
- DWORD **fsize**
- DWORD **sclust**
- DWORD **clust**
- DWORD **dsect**
- DWORD **dir_sect**
- BYTE * **dir_ptr**
- UINT **lockid**
- BYTE **buf** [_MAX_SS]

The documentation for this struct was generated from the following file:

- ff.h

5.6 FILESEM Struct Reference

Public Attributes

- FATFS * **fs**
- DWORD **clu**
- WORD **idx**
- WORD **ctr**

The documentation for this struct was generated from the following file:

- ff.c

5.7 FILINFO Struct Reference

Public Attributes

- DWORD **fsize**
- WORD **fdate**
- WORD **ftime**
- BYTE **fattrib**
- TCHAR **fname** [13]
- TCHAR * **lfname**
- UINT **lfsize**

The documentation for this struct was generated from the following file:

- ff.h

5.8 GameInfo Struct Reference

Models Game State.

`#include <main.h>`

Public Attributes

- Bird * birdy
- queue * que
- Coin * coin
- PlayerLevel * playerLevel
- bool alive
- int difficulty
- int frameCount
- int score
- int pipeGap
- Score highScore
- int pipeDistance
- Score scores [4]
- int **birdType**

5.8.1 Detailed Description

Models Game State.

Stores all game information regarding the current state of the game.

5.8.2 Member Data Documentation

5.8.2.1 alive

```
bool GameInfo::alive
```

Boolean check to see if the game is still playing

5.8.2.2 birdy

```
Bird* GameInfo::birdy
```

Pointer to bird in play

5.8.2.3 coin

```
Coin* GameInfo::coin
```

Pointer to coin in play

5.8.2.4 difficulty

```
int GameInfo::difficulty
```

Difficulty rating (0-2)

5.8.2.5 frameCount

```
int GameInfo::frameCount
```

Counter for frames

5.8.2.6 highScore

```
Score GameInfo::highScore
```

Highest score achieved in game state

5.8.2.7 pipeDistance

```
int GameInfo::pipeDistance
```

Distance between concurrent pipes

5.8.2.8 pipeGap

```
int GameInfo::pipeGap
```

Distance between top and bottom pipe

5.8.2.9 playerLevel

```
PlayerLevel* GameInfo::playerLevel
```

Pointer to player level in play

5.8.2.10 que

```
queue* GameInfo::que
```

Pointer to queue containing all pipes

5.8.2.11 score

```
int GameInfo::score
```

Current score of the game

5.8.2.12 scores

```
Score GameInfo::scores[4]
```

Saves n most current scores

The documentation for this struct was generated from the following file:

- main.h

5.9 Pipe Struct Reference

Models a pipe.

```
#include <main.h>
```

Public Attributes

- bool up
- int x
- int speed
- int topY
- int bottomY
- struct Pipe * next

5.9.1 Detailed Description

Models a pipe.

Structure to model all information needed to draw a pipe. a pipe will be added to the pipeQueue

5.9.2 Member Data Documentation

5.9.2.1 bottomY

```
int Pipe::bottomY
```

Y position of the bottom half of the pipe

5.9.2.2 next

```
struct Pipe* Pipe::next
```

Pointer to the following piper

5.9.2.3 speed

```
int Pipe::speed
```

Speed the pipe moves at

5.9.2.4 topY

```
int Pipe::topY
```

Y position of the top half of the pipe

5.9.2.5 up

```
bool Pipe::up
```

Weather the pipe is moving up or down (insane difficulty)

5.9.2.6 x

```
int Pipe::x
```

x co-ordinate

The documentation for this struct was generated from the following file:

- main.h

5.10 PlayerLevel Struct Reference

Tracks the user's current level.

```
#include <main.h>
```

Public Attributes

- int playerLevel
- int currentXp
- int requiredXp

5.10.1 Detailed Description

Tracks the user's current level.

Stores all game information regarding the current level of the player.

5.10.2 Member Data Documentation

5.10.2.1 currentXp

```
int PlayerLevel::currentXp
```

amount of xp on current level

5.10.2.2 playerLevel

```
int PlayerLevel::playerLevel
```

Stores level progress for user

5.10.2.3 requiredXp

```
int PlayerLevel::requiredXp
```

amount of xp till next level

The documentation for this struct was generated from the following file:

- main.h

5.11 putbuff Struct Reference

Public Attributes

- FIL * **fp**
- int **idx**
- int **nchr**
- BYTE **buf** [64]

The documentation for this struct was generated from the following file:

- ff.c

5.12 queue Struct Reference

Queue of pipes.

```
#include <main.h>
```

Public Attributes

- Pipe * head
- Pipe * tail

5.12.1 Detailed Description

Queue of pipes.

Creates a queue of pipes that is used by the game state

5.12.2 Member Data Documentation

5.12.2.1 head

```
Pipe* queue::head
```

Points to the head pipe of the queue

5.12.2.2 tail

```
Pipe* queue::tail
```

Points to the tail pipe of the queue

The documentation for this struct was generated from the following file:

- main.h

5.13 Rectangle Struct Reference

Models a prectangle.

```
#include <main.h>
```

Public Attributes

- int Xpos
- int Ypos
- int Xpos2
- int Ypos2

5.13.1 Detailed Description

Models a prectangle.

Stores all the points of a rectangle

5.13.2 Member Data Documentation

5.13.2.1 Xpos

```
int Rectangle::Xpos
```

Top left corner

5.13.2.2 Xpos2

```
int Rectangle::Xpos2
```

Top right corner

5.13.2.3 Ypos

```
int Rectangle::Ypos
```

Bottom left corner

5.13.2.4 Ypos2

```
int Rectangle::Ypos2
```

Bottom right corner

The documentation for this struct was generated from the following file:

- main.h

5.14 Score Struct Reference

All score information.

```
#include <main.h>
```

Public Attributes

- int score
- int difficulty

5.14.1 Detailed Description

All score information.

Stores all information regarding score

5.14.2 Member Data Documentation

5.14.2.1 difficulty

```
int Score::difficulty
```

Level of difficulty when score was achieved

5.14.2.2 score

```
int Score::score
```

Score achieved

The documentation for this struct was generated from the following file:

- main.h

Chapter 6

File Documentation

6.1 Absurd.c File Reference

bitmap image of Absurd button

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmAbsurd**

6.1.1 Detailed Description

bitmap image of Absurd button

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.1.2 Variable Documentation

6.1.2.1 bmAbsurd

```
GUI_CONST_STORAGE GUI_BITMAP bmAbsurd
```

Initial value:

```
= {
  80,
  50,
  320,
  32,
  (unsigned char *)_acAbsurd,
  NULL,
  GUI_DRAW_BMP8888
}
```

6.2 Background.c File Reference

bitmap image of background

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmBackground**

6.2.1 Detailed Description

bitmap image of background

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.2.2 Variable Documentation

6.2.2.1 bmBackground

```
GUI_CONST_STORAGE GUI_BITMAP bmBackground
```

Initial value:

```
= {
  490,
  280,
  490,
  8,
  _acBackground,
  &_PalBackground
}
```

6.3 BlueBird.c File Reference

bitmap image of bird player

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmBlueBird**

6.3.1 Detailed Description

bitmap image of bird player

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.3.2 Variable Documentation

6.3.2.1 bmBlueBird

```
GUI_CONST_STORAGE GUI_BITMAP bmBlueBird
```

Initial value:

```
= {
  34,
  24,
  136,
  32,
  (unsigned char *)_acBlueBird,
  NULL,
  GUI_DRAW_BMP8888
}
```

6.4 Coin.c File Reference

bitmap image of arrow

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmCoin**

6.4.1 Detailed Description

bitmap image of arrow

bitmap image of Logo 2

bitmap image of logo

bitmap image of King Bird

bitmap image of easter egg

bitmap image of cowboy bird

bitmap image of coin

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.4.2 Variable Documentation

6.4.2.1 bmCoin

```
GUI_CONST_STORAGE GUI_BITMAP bmCoin
```

Initial value:

```
= {
  25,
  25,
  100,
  32,
  (unsigned char *)_acCoin,
  NULL,
  GUI_DRAW_BMP8888
}
```

6.5 Easy.c File Reference

bitmap image of easy button

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmEasy**

6.5.1 Detailed Description

bitmap image of easy button

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.5.2 Variable Documentation

6.5.2.1 bmEasy

```
GUI_CONST_STORAGE GUI_BITMAP bmEasy
```

Initial value:

```
= {
  80,
  50,
  320,
  32,
  (unsigned char *)_acEasy,
  NULL,
  GUI_DRAW_BMP8888
}
```

6.6 flappyBird.c File Reference

File containing all methods to update the gamestate.

```
#include "stm32f7xx_hal.h"
#include "stm32746g_discovery_sdram.h"
#include "RTE_Components.h"
#include "GUI.h"
#include <stdlib.h>
#include "main.h"
#include "Progbar.h"
#include <math.h>
```

Functions

- void setupGameInfo ()
  SetupGameInfo, sets up game based on difficulty.
- void setDifficulty (int choice)
  Sets game difficulty.
- void updateScores ()
  Updates the scoreboard.
- void createPipeQueue ()
  Assigns a pipe queue to the game state.
- bool hits (Pipe *p)
  Detects collisions.
- void initPipes ()
  Initialises a pipe.
- void updateAllPipes ()
  Updates all pipes in the queue.
- bool isbirdAlive ()
  If the bird is alive or not.

- void initBird ()
  Initialises a bird.
- void upBirdy (void)
  Moves the bird up.
- void updateBirdy (void)
  Updates the birds state.
- void initCoin ()
  Initialises a coin.
- void updateCoin ()
  Updates coin positioning.
- void drawProgress ()
  Draws progress circles.
- void drawBirdChoice ()
  Draws the bird selector.
- void drawHighscores (void *pData)
  Displays scores and difficulty associated with that score.
- void displayLeaderboard (TOUCH_STATE tsc_state, GUI_RECT Rect)
  Creates a window to display scores and select difficulty.
- void initPregame (TOUCH_STATE tsc_state, GUI_RECT Rect)
  Initilise pregame.
- void drawEverything (GUI_RECT Rect)
  Allocates memory for drawing elements.

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmCowboyBirdy**
- GUI_CONST_STORAGE GUI_BITMAP **bmBlueBird**
- GUI_CONST_STORAGE GUI_BITMAP **bmKingBirdy**
- GUI_CONST_STORAGE GUI_BITMAP **bmBackground**
- GUI_CONST_STORAGE GUI_BITMAP **bmCoin**
- GUI_CONST_STORAGE GUI_BITMAP **bmPregame**
- GUI_CONST_STORAGE GUI_BITMAP **bmEasy**
- GUI_CONST_STORAGE GUI_BITMAP **bmHard**
- GUI_CONST_STORAGE GUI_BITMAP **bmAbsurd**
- GUI_CONST_STORAGE GUI_BITMAP **bmArrow**
- GUI_CONST_STORAGE GUI_BITMAP **bmlogo2**
- GUI_CONST_STORAGE GUI_BITMAP **bmEgg**
- GameInfo **gameInfo**
- int **turn** = 0
- bool **xpSet** = false

6.6.1 Detailed Description

File containing all methods to update the gamestate.

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.7 Hard.c File Reference

bitmap image of Hard button

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmHard**

6.7.1 Detailed Description

bitmap image of Hard button

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.7.2 Variable Documentation

6.7.2.1 bmHard

```
GUI_CONST_STORAGE GUI_BITMAP bmHard
```

Initial value:

```
= {
  80,
  50,
  320,
  32,
  (unsigned char *)_acHard,
  NULL,
  GUI_DRAW_BMP8888
}
```

6.8 main.c File Reference

File containing all setup methods to get the game initialised.

```
#include "stm32f7xx_hal.h"
#include "stm32746g_discovery_sdram.h"
#include "RTE_Components.h"
#include "GUI.h"
#include <stdlib.h>
#include "main.h"
#include "Board_Touch.h"
```

Functions

- void initGame ()

  Initialise the game state.
- void MainTask (void)

  Task container.
- int main (void)

  Main.

Variables

- FILE * **file**

6.8.1 Detailed Description

File containing all setup methods to get the game initialised.

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.9 main.h File Reference

header file containing all methods and structs

```
#include <stdbool.h>
#include "Board_Touch.h"
```

Classes

- struct Rectangle
  Models a prectangle.
- struct Pipe
  Models a pipe.
- struct queue
  Queue of pipes.
- struct Bird
  Models a bird.
- struct Coin
  Models a coin.
- struct Score
  All score information.
- struct PlayerLevel
  Tracks the user's current level.
- struct GameInfo
  Models Game State.

Typedefs

- typedef struct Pipe Pipe
  Models a pipe.
- typedef struct queue queue
  Queue of pipes.
- typedef struct Bird Bird
  Models a bird.
- typedef struct Coin Coin
  Models a coin.
- typedef struct Score Score
  All score information.
- typedef struct PlayerLevel PlayerLevel
  Tracks the user's current level.
- typedef struct GameInfo GameInfo
  Models Game State.

Functions

- void queueDestroy (struct queue *q)
  Removes entire queue and frees memory.
- int queueEmpty (const struct queue *q)
  Checks if the queue is empty.
- void updateAllPipes (void)
  Updates all pipes in the queue.
- void deq (queue *q)
  Removes pipe at the head of the queue.
- void updatePipes (queue *q)
  Updates pipe positions.
- void **erasePipes** (queue *q)
- void enq (queue *q, Pipe *p)

Push pipe to queue.

- queue * queueCreate (void)

  Creates empty queue structure.
- Pipe * **getList** (int position)
- bool isOffScreen (queue *q)

  Checks if the queue is off screen.
- void initGame (void)

  Initialise the game state.
- void initBird (void)

  Initialises a bird.
- void initCoin (void)

  Initialises a coin.
- void setupGameInfo (void)

  SetupGameInfo, sets up game based on difficulty.
- void initPipes (void)

  Initialises a pipe.
- void displayLeaderboard (TOUCH_STATE tsc_state, GUI_RECT Rect)

  Creates a window to display scores and select difficulty.
- void initPregame (TOUCH_STATE tsc_state, GUI_RECT Rect)

  Initilise pregame.
- bool isbirdAlive (void)

  If the bird is alive or not.
- void updateCoin (void)

  Updates coin positioning.
- void updateBirdy (void)

  Updates the birds state.
- void drawEverything (GUI_RECT Rect)

  Allocates memory for drawing elements.
- void createPipeQueue (void)

  Assigns a pipe queue to the game state.
- void setDifficulty (int choice)

  Sets game difficulty.

6.9.1 Detailed Description

header file containing all methods and structs

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.9.2 Typedef Documentation

6.9.2.1 Bird

```
typedef struct Bird Bird
```

Models a bird.

Structure to model all information needed to draw a bird.

6.9.2.2 Coin

```
typedef struct Coin Coin
```

Models a coin.

Structure to model all information needed to draw a coin.

6.9.2.3 GameInfo

```
typedef struct GameInfo GameInfo
```

Models Game State.

Stores all game information regarding the current state of the game.

6.9.2.4 Pipe

```
typedef struct Pipe Pipe
```

Models a pipe.

Structure to model all information needed to draw a pipe. a pipe will be added to the pipeQueue

6.9.2.5 PlayerLevel

```
typedef struct PlayerLevel PlayerLevel
```

Tracks the user's current level.

Stores all game information regarding the current level of the player.

6.9.2.6 queue

```
typedef struct queue queue
```

Queue of pipes.

Creates a queue of pipes that is used by the game state

6.9.2.7 Score

```
typedef struct Score Score
```

All score information.

Stores all information regarding score

6.10 Pregame.c File Reference

bitmap image of pregame background

```
#include <stdlib.h>
#include "GUI.h"
```

Macros

- #define **GUI_CONST_STORAGE** const

Variables

- GUI_CONST_STORAGE GUI_BITMAP **bmPregame**

6.10.1 Detailed Description

bitmap image of pregame background

Author

Luke Garrigan, Shane Sturgeon

Date

17 March 2017

6.10.2 Variable Documentation

6.10.2.1 bmPregame

```
GUI_CONST_STORAGE GUI_BITMAP bmPregame
```

Initial value:

```
= {
  120,
  141,
  480,
  32,
  (unsigned char *)_acPregame,
  NULL,
  GUI_DRAW_BMP8888
}
```

Index