Estado final do projeto.

Como produto final, temos uma programa capaz de correr através da passagem por um menu do jogo, controlado pelo teclado, e o jogo em si, sendo que o desenho das paredes é controlado pelo rato, pelo que dentro do próprio jogo o teclado serve o propósito de parar a sua normal execução através da tecla “esc”.

Sumariando as partes a que nos propusemos e que foram implementadas temos:

- Linhas a ser desenhadas tanto na horizontal como na vertical, sendo que estas começam por ser desenhadas a partir de um ponto focal (o cursor do rato) e só param quando atingem uma outra parede;

- O jogador perde quando uma bola atravessa uma parede a ser desenhada ou quando o contador do tempo chega a 0 e o jogador ainda não atingiu a área pretendida;

- Quando uma linha acaba de ser desenhada, os limites dentro dos quais a bola pode circular são alterados;

- A espaço livre da bola é atualizado sempre que os limites sao alterados;

- As bolas fazem uma reflexão completa nas paredes;

- Um contador a mostrar quanto tempo é que o jogador ainda tem;

- O cursor do rato torna-se fixo enqunato se mantém premido o botão da esquerda, aparacendo um indicador com a direção para onde a parede vai ser desenhada;

Partes que nao foram concluidas:

-Porta de série está implementada (código todo desenvolvido) mas não implementado no jogo (sem funcionalidade multiplayer);

Instruções de execução

Após o início do programa, o jogador depara – se com 2 opções, start ou exit. Para circular entre elas deve usar os números do teclado, sendo um número correspondente a uma opção. Entrando no jogo, o jogador tem 3 opções, ou sai do jogo através da tecla “esc”, ou perde desenhando uma parede de forma a que a bola choque contra esta enquanto esta a ser desenhada, ou ganha o jogo, aprisionando a bola no espaço de tamanho mínimo indicado no incío do jogo.

Descrição da Arquitectura do programa

boolType.h – enum type de modo a tornar o código mais legível;

ClockClass.h – módulo destinado a gerir o RTC e o timer (Inicialização, Handle de interrupts, geração de eventos do RTC após uma interrupção);

CoordStruct.h – struct para representar coordenadas;

DispatcherClass.h – módulo destinado a fazer o subscribe de todos os periféricos, fazendo gestão dos hook_ids, das IRQ lines de cada perifério e faz handle de todas as interrupções). Contém uma enum type devices_t para representar os periféricos. Por cada interrupção do timer, um frame é desenhado no ecrã.

GameLogicClass.h – nesta class são processados todos os eventos:

- Eventos do RTC: sempre que é gerada uma interrupção do tipo Update Ended, o contador que é mostrado no jogo decresce uma unidade.

- Eventos do keyboard: sempre que é primida uma tecla é gerado um evento com o makecode correspondente, que depois é processado.

- Eventos do mouse: se o botao esquerdo for primido, o rato passa a um estado de "lock", em que é possivel escolher a direção (horizontal ou vertical) para cosntruir uma parede. É feita a distinção entre o evento mouse_down (quando um botao é primido) e o evento mouse_up (quando o botão deixa de ser primido)

GraphicsClass.h – Inicializa o jogo em modo gráfico direto. É utilizado o double buffering, ou seja, tudo é desenhado primeiro num backBuffer e apenas quando a função GraphicsFrame é chamada é que tudo é desenhado no ecrã de uma só vez. As paredes são desenhadas num buffer à parte, por uma questão de implementação da lógica do jogo. Também contém funções que desenham os sprites, paredes e números no backBuffer.

InputClass.h – Está encarregue de inicializar os modulos do mouse e do keyboard. Processa interrupções de ambos os periféricos e gera eventos. Também é processada toda a lógica do mouse lock e cálculo da direção em que as walls e o sprite "Triangle" devem ser desenhados.

KeyboardClass.h – Contém funções para comunicar com o teclado do computador semelhantes às do laboratório.

ListClass.h – Módulo destinado a criar lista duplamente ligadas, tendo como "inspiração" as listas de c++. Pode assim simular queues e stacks. Muito útil para a geração de eventos do RTC, Keyboard e Mouse.

MenuClass.h – Módulo encarregue da gestão do menu que é apresentado no inicio do jogo. Contém sprites para desenhar as entradas do menu "automaticamente" e processa eventos gerados pelo keyboard (número das opções e a tecla escape para sair do programa). Comunica

com a GraphicsClass, InputClass e GameLogic para entrar no jogo propriamente dito e para regressar ao jogo.

MenuSprites.h – contém os sprites correspondentes às entradas do menu.

MouseClass.h – Módulo que faz a comunicação com o rato. Tem funções semelhantes às do laboratório. O MouseEnableMovementReport é o caso de uma função criada como propósito de debug.

numbers.h – contém os sprites dos algarismos.

RTCClass.h – Módulo que comunica com o RTC. Contém funções semelhantes às do laboratório e funções para ativar os três tipos de interrupções. É utilizado asembly para comunicar com os registos – funções **RTCReadReg** e **RTCWriteReg.**

SerialPortClass.h – Módulo que comunica com a porta de série. Contém funções semelhantes às do lab. Funções em Assembly – **`SerialPortReadReg` e `SerialPortWriteReg`.**

SpriteClass.h – Class que representa Sprites. Gere informação como a posição, movimento, limites e dimensões. Podem ser inicializados Sprites a partir de XPM's. A função para ler a partir de BMP's foi criada, mas nunca utilizada.

SystemClass.h – Módulo que inicializa todo o programa. No SystemRun é chamado em primeiro lugar o DispatcherRun que verifica que se existe alguma interrupção pendente e de seguida é processada toda a logica do jogo.

TimerClass.h – Faz a comunicação com timer do pc. Funções semelhantes à do laboratório. Está configurado para gerar interrupções a 60Hz.

VBEModeInfoStruct.h – struct utilizada na inicializaçao da VBE na GraphicsClass.

Outros Aspestos de Implementação

O código está todo estruturado em classes e foi criado de modo a pensar na sua futura utilização. Existe um número razoável de funções que não utilizadas, mas que existem de modo a que seja mais fácil o debug caso seja necessário. Criou-se vários enum types de modo a que o código fosse mais fácil de compreender. Se se pretender utilizar outros eventos do mouse ou do keyboard, a sua implementação nesta fase final do programa é muito fácil, assim como se se quisesse gerar.

A DispatcherClass permite a reutilização do código do subscribe e unsubscribe para todos os periféricos. Todas as interrupções são detetadas no DispatcherRun.

São utilizadas listas para simular uma queue de eventos para o Mouse, para o Keyboard e para o RTC. Estes eventos são gerados na InputClass e na ClockClass, e processados na GameLogicClass (e MenuClass também, no caso dos do keyboard).

Foram utilizados sprites para desenhar os algarismos e entradas dos menus, assim como os triangulos que indicam a direção em que as paredes são desenhadas.

O programa funciona em modo de texto para efeitos de debug (basta modificar um dos parâmetro de SystemInitialize de true para false).

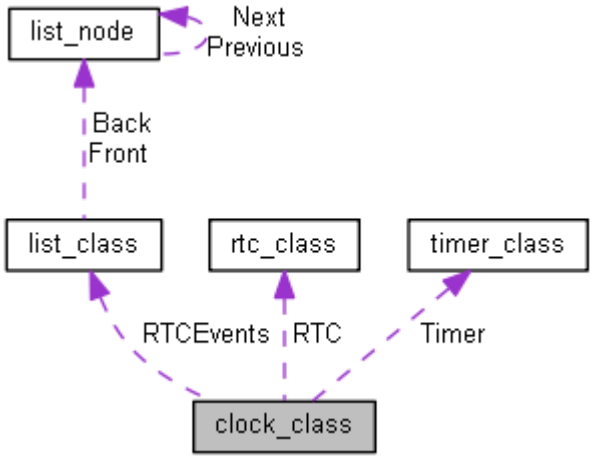
Foram implementadas algumas funções em assembly para comunicar com os registos na SerialPortClass e na RTCClass.

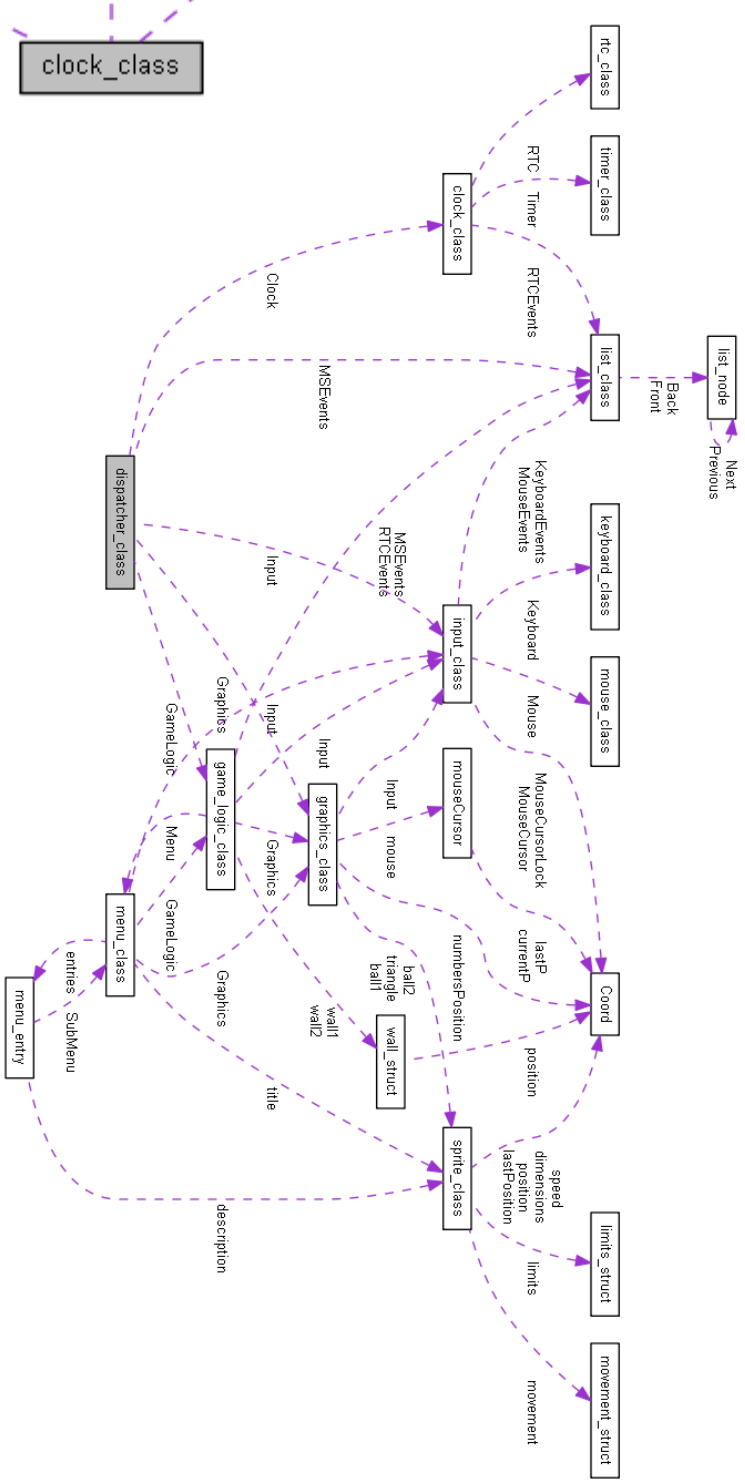
O código dos laboratórios foi competamente revisto e melhorado de modo integrar as classes.

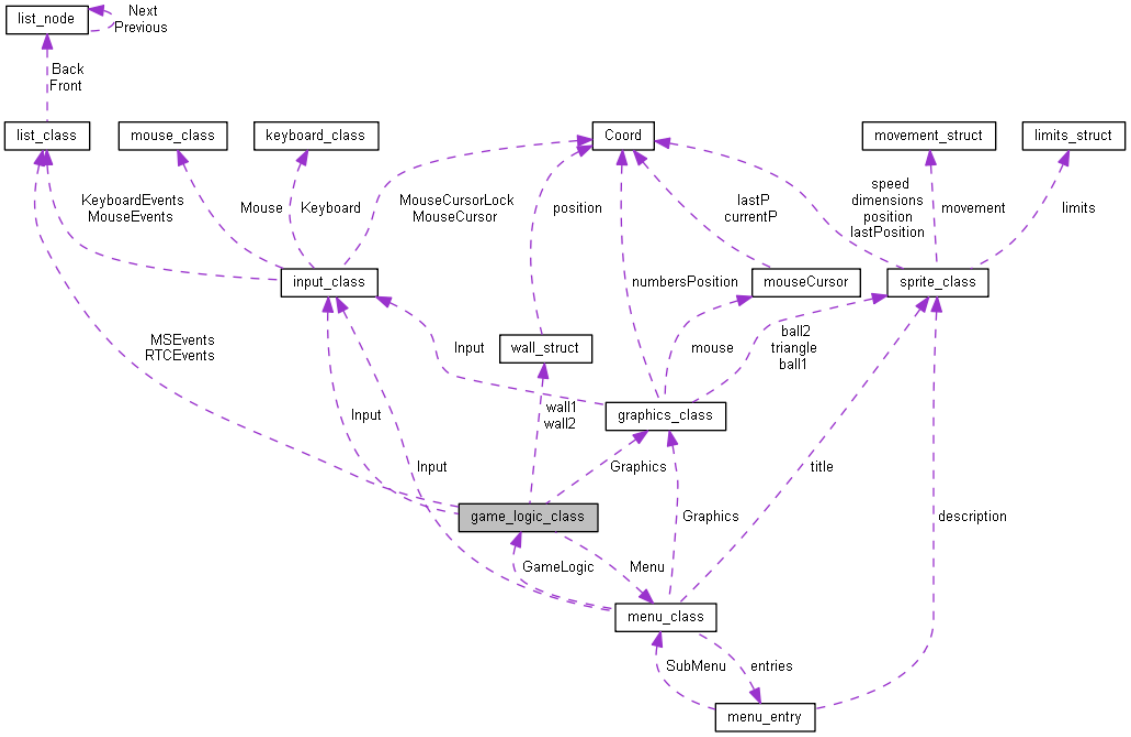
O modo VBE utilizado é o 0x117 (1024x768, 16-bits).

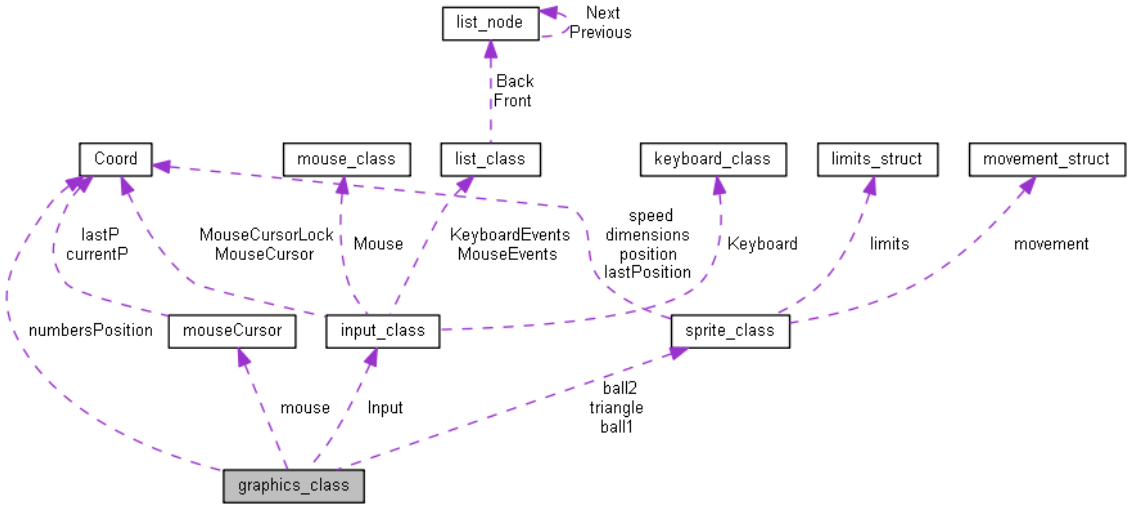
A porta de série foi completamente revista e implementada em forma de class no código. No entanto, não foi implementado o modo multiplayer.

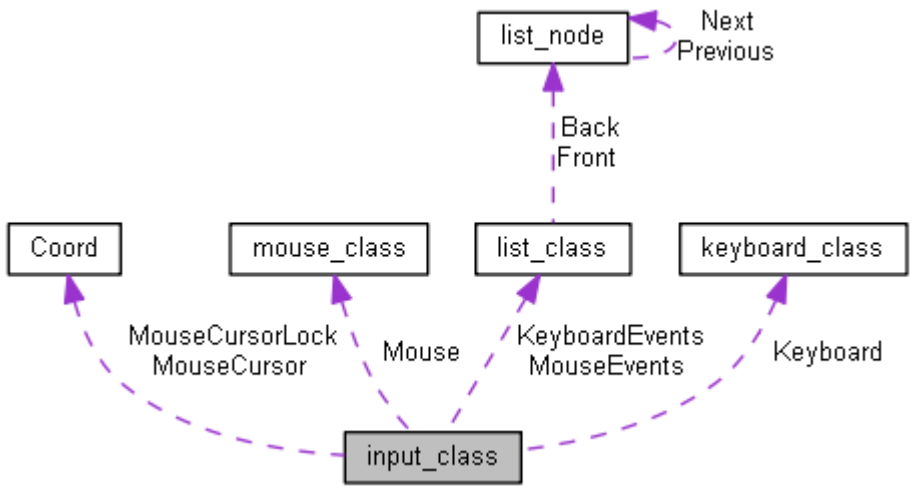
Function Diagrams

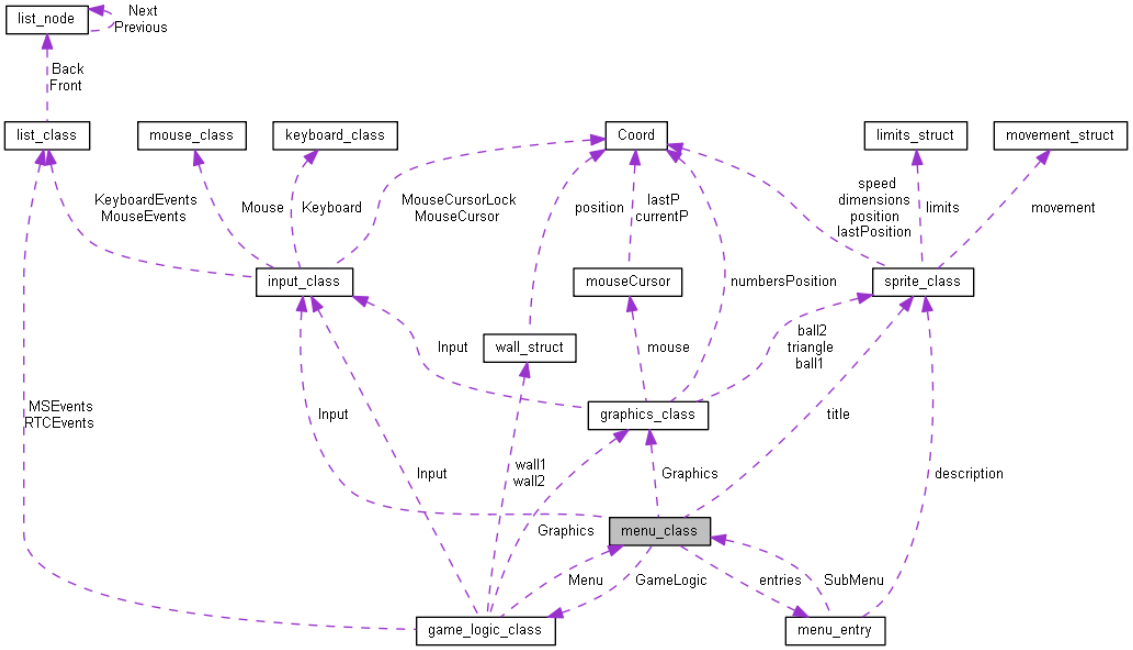


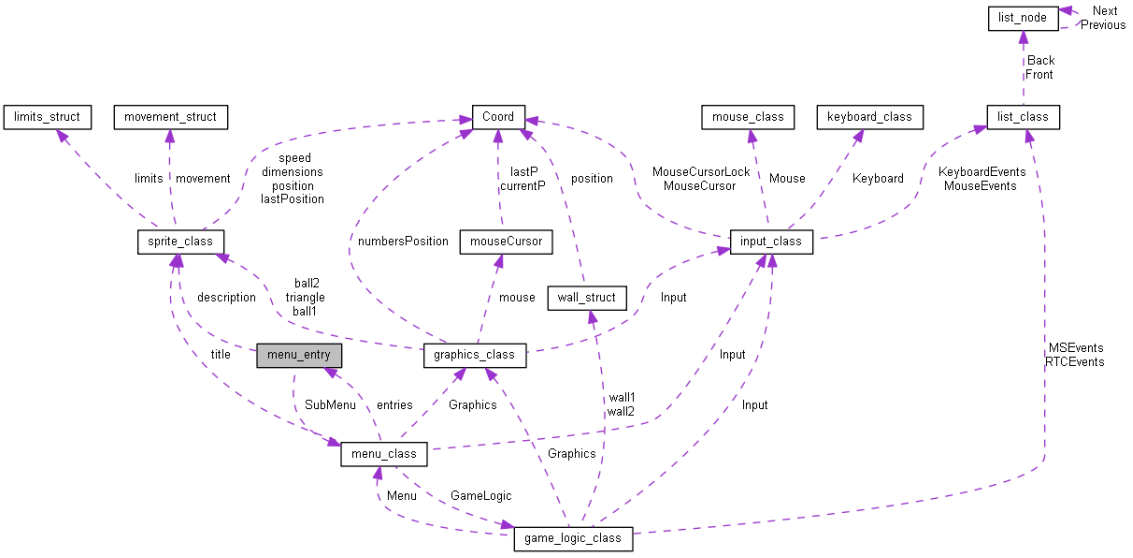


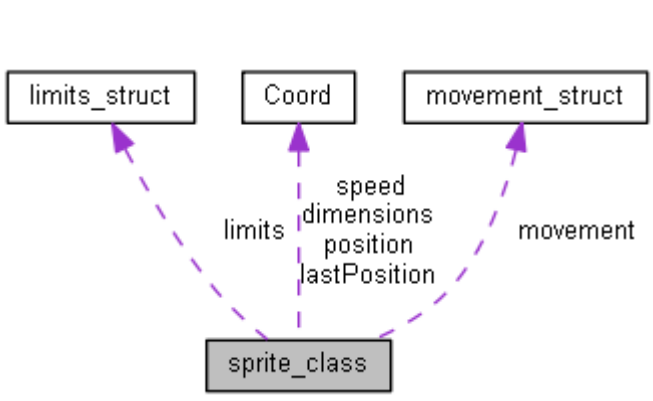

list_node
Next
Previous
Back
Front
list_class
mouse_class
keyboard_class
Coord
movement_struct
limits_struct
KeyboardEvents
MouseEvents
Mouse
Keyboard
MouseCursorLock
MouseCursor
position
lastP
currentP
speed
dimensions
position
lastPosition
movement
limits
input_class
numbersPosition
mouseCursor
sprite_class
MSEvents
RTCEvents
Input
wall_struct
mouse
ball2
triangle
ball1
wall1
wall2
graphics_class
Input
Input
Graphics
title
game_logic_class
Graphics
description
GameLogic
Menu
menu_class
SubMenu
entries
menu_entry

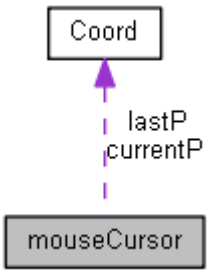

list_node
Next
Previous
Back
Front
Coord
mouse_class
list_class
keyboard_class
limits_struct
movement_struct
lastP
currentP
MouseCursorLock
MouseCursor
Mouse
KeyboardEvents
MouseEvents
speed
dimensions
position
lastPosition
Keyboard
limits
movement
numbersPosition
mouseCursor
input_class
sprite_class
mouse
Input
ball2
triangle
ball1
graphics_class

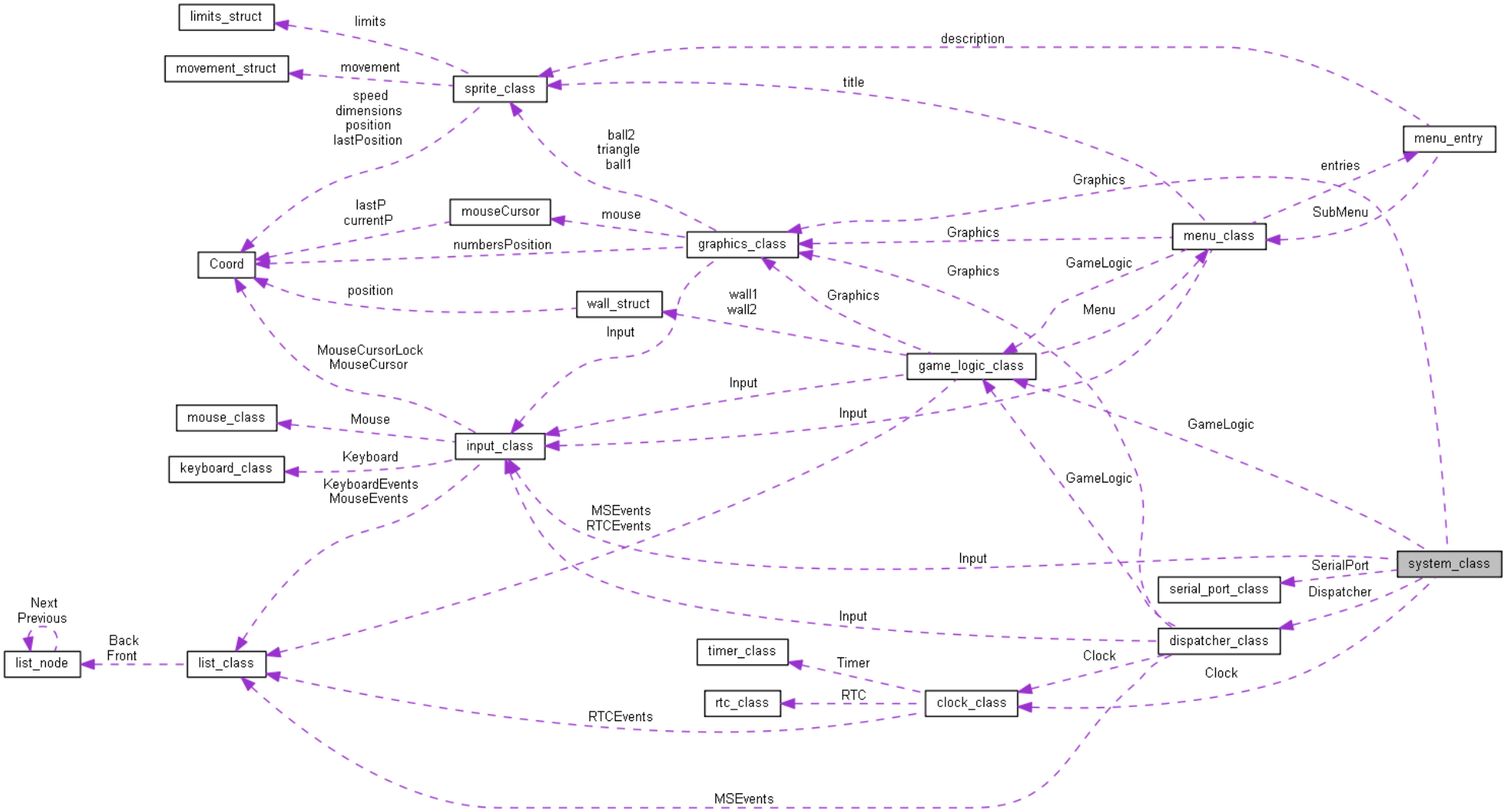

list_node
Next
Previous
Back
Front
Coord
mouse_class
list_class
keyboard_class
MouseCursorLock
MouseCursor
Mouse
KeyboardEvents
MouseEvents
Keyboard
input_class


list_node
Next
Previous
Back
Front
list_class
mouse_class
keyboard_class
Coord
limits_struct
movement_struct
KeyboardEvents
MouseEvents
Mouse
Keyboard
MouseCursorLock
MouseCursor
position
lastP
currentP
speed
dimensions
position
lastPosition
limits
movement
input_class
mouseCursor
numbersPosition
sprite_class
Input
wall_struct
mouse
ball2
triangle
ball1
MSEvents
RTCEvents
graphics_class
title
wall1
wall2
Graphics
description
menu_class
Menu
GameLogic
entries
SubMenu
game_logic_class
menu_entry


list_node
Next
Previous
Back
Front
limits_struct
movement_struct
Coord
mouse_class
keyboard_class
list_class
limits
movement
speed
dimensions
position
lastPosition
lastP
currentP
position
MouseCursorLock
MouseCursor
Mouse
Keyboard
KeyboardEvents
MouseEvents
sprite_class
numbersPosition
mouseCursor
input_class
description
ball2
triangle
ball1
mouse
wall_struct
Input
title
menu_entry
graphics_class
MSEvents
RTCEvents
SubMenu
entries
Graphics
wall1
wall2
menu_class
Menu
GameLogic
game_logic_class


limits_struct
Coord
movement_struct
limits
speed
dimensions
position
lastPosition
movement
sprite_class


Coord
lastP
currentP
mouseCursor


limits_struct
limits
movement_struct
movement
sprite_class
speed
dimensions
position
lastPosition
description
title
menu_entry
entries
ball2
triangle
ball1
Graphics
lastP
currentP
mouseCursor
mouse
graphics_class
menu_class
SubMenu
Graphics
numbersPosition
Coord
Graphics
GameLogic
position
wall_struct
wall1
wall2
Graphics
Menu
Input
MouseCursorLock
MouseCursor
game_logic_class
Input
mouse_class
Mouse
Input
GameLogic
input_class
keyboard_class
Keyboard
KeyboardEvents
MouseEvents
GameLogic
MSEvents
RTCEvents
Input
SerialPort
system_class
serial_port_class
Dispatcher
Next
Previous
Input
list_node
Back
Front
list_class
timer_class
Timer
Clock
dispatcher_class
Clock
rtc_class
RTC
clock_class
RTCEvents
MSEvents

AutoAvaliação

João Pedro Miranda Maia

Código da sua responsabilidade : Criação das classes, implementação dos periféricos, Dispatcher, gestão de eventos e Menu.

Documentação : Explicação das classes e aspetos relevantes da implementação.

Peso relativo da participação: 50%;

Peso relativo da contribuição para o projeto final : 60%

José Paulo Santos Oliveira

Código da sua responsabilidade: Gestão do jogo, placa gráfica e sprites;

Documentação : Estado final do projeto, instruções de execução e criação de doxyfile;

Peso relativo da participação: 50%;

Peso relativo da contribuição para o projeto final : 40%