



Ministério da Educação e Ciência

## Escola Secundária de Avelar Brotero

Ano letivo 2013 / 2014

**Curso Profissional de Técnico  
de  
Gestão e Programação de Sistemas Informáticos**

# **Projeto da Prova de Aptidão Profissional**

### *Tema*

. Super Mario Bros em c++ .

15 de Outubro de 2013

<b>Nome do Autor:</b>	Rodrigo Reis
<b>Ano / Turma / Número</b>	12
<b>Professor Orientador:</b>	José Carlos Martins

**Curso Profissional de Técnico  
de  
Gestão e Programação de Sistemas Informáticos  
Projeto  
Da  
Prova de Aptidão Profissional**



# Índice

Título .....	7
Descrição do projeto .....	7
Síntese.....	7
Fases da criação do projeto.....	7
Protótipo .....	8
Aquisição do material.....	9
Preparação do ambiente de trabalho.....	9
Criação do jogo.....	10
EntryPoint.....	10
Makefile.....	11
Core .....	12
Game.....	12
Timer .....	15
Drawings.....	16
Input.....	19
SoundManager.....	21
Sprite.....	23
Screens.....	27
ScreenManager .....	27
BaseScreen.....	29
TitleScreen.....	30
OptionsScreen.....	32
ScoreBoardSelection .....	34
Scoreboard .....	36
NewGameScreen .....	38
NextLevelScreen .....	41
GameScreen.....	43
SubmitScoreScreen.....	45
GUI .....	47
CustomFont .....	47
Botão.....	48
Textbox.....	49
Hud .....	52
Table .....	55
Entities .....	57
Ent.....	57
LivingEnt .....	60
MysteryBox .....	63
Goomba .....	66
Controller.....	67
Objetivos do projeto .....	70
Interesse e aplicabilidade do projeto .....	70
Disciplina envolvidas no projeto .....	70
Saberes e competências profissionais.....	70
Fases temporais do projeto .....	70
Horas previstas na implementação do projeto .....	71
Recursos Humanos envolvidos.....	71
Recursos Materiais envolvidos.....	71
Custos Estimados do projeto .....	71

Reflexão sobre a exequibilidade do projeto .....	71
Parceria .....	71
Bibliografia.....	72

## Índice de figuras

<i>Figura 1 - Protótipo em python .....</i>	8
<i>Figura 2 - Função main .....</i>	10
<i>Figura 3 - MakeFile .....</i>	11
<i>Figura 4 - Definições da classe Game .....</i>	12
<i>Figura 5 - Código da classe Game.....</i>	13
<i>Figura 6 - Código da classe Game.....</i>	14
<i>Figura 7 - Código da classe Game.....</i>	14
<i>Figura 8 - Definições da classe Timer .....</i>	15
<i>Figura 9 - Código da classe Timer.....</i>	15
<i>Figura 10 - Definição da funções do ficheiro Drawings.h.....</i>	16
<i>Figura 11 - Código do ficheiro Drawings.h.....</i>	17
<i>Figura 12 – Código do ficheiro Drawings.cpp .....</i>	18
<i>Figura 13 - Definições do ficheiro Input.h.....</i>	19
<i>Figura 14 - Código do Input .....</i>	20
<i>Figura 15 - Código do Input .....</i>	20
<i>Figura 16 - Definições SoundManager .....</i>	21
<i>Figura 17 - Código do classe SoundManager .....</i>	22
<i>Figura 18 - Código da classe SoundManager .....</i>	22
<i>Figura 19 - Definições da classe Sprite .....</i>	23
<i>Figura 20 - Código da classe Sprite.....</i>	24
<i>Figura 21 - Código da classe Sprite.....</i>	25
<i>Figura 22 - Código da classe Sprite.....</i>	26
<i>Figura 23 - Definições do SoundManager .....</i>	27
<i>Figura 24 - Código da classe ScreenManager .....</i>	28
<i>Figura 25 - Definições da classe BaseScreen .....</i>	29
<i>Figura 26 - Imagem da tela inicial.....</i>	30
<i>Figura 27 - Definições da classe TitleScreen.....</i>	30
<i>Figura 28 - Código da TitleScreen.....</i>	31
<i>Figura 29 - Imagem da tela de opções .....</i>	32
<i>Figura 30 - Definições da classe OptionsScreen .....</i>	32
<i>Figura 31 - Código da classe OptionsScreen.....</i>	33
<i>Figura 32 - Imagem da tela de seleção de scoreboard .....</i>	34
<i>Figura 33 - Definições da classe ScoreBoardSelection .....</i>	34
<i>Figura 34 - Código da tela ScoreBoard Selection .....</i>	35
<i>Figura 35 - Imagem da tela de pontuação .....</i>	36
<i>Figura 36 - Definições da classe ScoreBoard .....</i>	36
<i>Figura 37 - Código da classe ScoreBoard .....</i>	37
<i>Figura 38 - Código da classe ScoreBoard .....</i>	37
<i>Figura 39 - Imagem da tela NewGameScreen .....</i>	38
<i>Figura 40 - Definições da classe NewGameScreen .....</i>	38
<i>Figura 41 - Código da classe NewGameScreen.....</i>	39
<i>Figura 42 - Código da classe NewGameScreen.....</i>	40
<i>Figura 43 - Definições da classe NextLevelScreen .....</i>	41
<i>Figura 44 - Código da classe NextLevelScreen .....</i>	42
<i>Figura 45 - Código da classe NextLevelScreen .....</i>	42
<i>Figura 46 - Imagem da tela do jogo.....</i>	43
<i>Figura 47 - Definições da classe GameScreen .....</i>	43
<i>Figura 48 - Código da classe GameScreen.....</i>	44

<i>Figura 49 - Código da classe GameScreen</i> .....	44
<i>Figura 50 - Definições da classe SubmitScore</i> .....	45
<i>Figura 51 - Código da classe SubmitScore</i> .....	46
<i>Figura 52 - Fonte normal do jogo</i> .....	47
<i>Figura 53 - Definições da classe CustomFont</i> .....	47
<i>Figura 54 - Código da classe CustomFont</i> .....	47
<i>Figura 55 - Botão normal</i> .....	48
<i>Figura 56 - Botão quando o mouse esta por cima</i> .....	48
<i>Figura 57 - Definições da classe botão</i> .....	48
<i>Figura 58 - Código da classe botão</i> .....	48
<i>Figura 59 - Caixa de texto</i> .....	49
<i>Figura 60 - Definições da classe TextBox</i> .....	49
<i>Figura 61 - Código da classe TextBox</i> .....	51
<i>Figura 62 - Imagem do Hud</i> .....	52
<i>Figura 63 - Definições da classe Hud</i> .....	52
<i>Figura 64 - Código da classe Hud</i> .....	53
<i>Figura 65 - Código da classe Hud</i> .....	54
<i>Figura 66 - Imagem de uma tabela criada</i> .....	55
<i>Figura 67 - Definições da classe Table</i> .....	55
<i>Figura 68 - Código da classe Table</i> .....	56
<i>Figura 69 - Definições da classe Ent</i> .....	58
<i>Figura 70 - Definições da classe Ent</i> .....	59
<i>Figura 71 - Código classe Ent</i> .....	59
<i>Figura 72 - Definições da classe LivingEnt</i> .....	60
<i>Figura 73 - Código da LivingEnt</i> .....	61
<i>Figura 74 - Código da classe LivingEnt</i> .....	62
<i>Figura 75 - Imagem da spriteSheet da MysteryBox</i> .....	63
<i>Figura 76 - Definições da classe MysteryBox</i> .....	63
<i>Figura 77 - Código da classe MysteryBox</i> .....	64
<i>Figura 78 - Código da classe MysteryBox</i> .....	65
<i>Figura 79 - Código da classe MysteryBox</i> .....	65

## Título

Super mario bros criado do zero usando SDL2 para poder desenhar na tela e c++ como linguagem de programação

## Descrição do projeto

### Síntese

Este projeto trata-se de uma recriação de super mario bros com gráficos super mario all stars.

Este jogo é feito em c ++ e para desenhar na tela eu uso SDL(Simple DirectMedia Layer).

Este jogo contém menu iniciar, tem também menu de opções onde dá para mudar tamanho da janela, quando se inicia o jogo ele irá carregar primeiro nível , onde tem hud que mostra o score,moedas,nível em que tem player que dá para movimentar usando WASD , tem níveis com blocos , inimigos e um fundo que se move relativamente ao personagem.

Todo o código encontra-se no meu repositório do github como o binário do jogo:  
<https://github.com/Rreis019/SuperMarioRecreation>

## Fases da criação do projeto

- Protótipo
- Planeamento
- Aquisição do material
- Preparação do ambiente trabalho
- Criação do jogo (câmeras,colisões,sprite loader,spritesheet, classes dos inimigos,players)

## Protótipo

Comecei por fazer protótipo na linguagem Python usando a biblioteca Pygame onde fiz a movimentação e a animação do personagem e as colisões, para saber se conseguia fazer jogo do zero mesmo.

Depois de feito o protótipo acabei por escolher o c++ para linguagem de programação, porque é mais rápido, sinto mais confortável com essa linguagem, para biblioteca gráfica usei SDL porque a mesma biblioteca que o pygame usava.

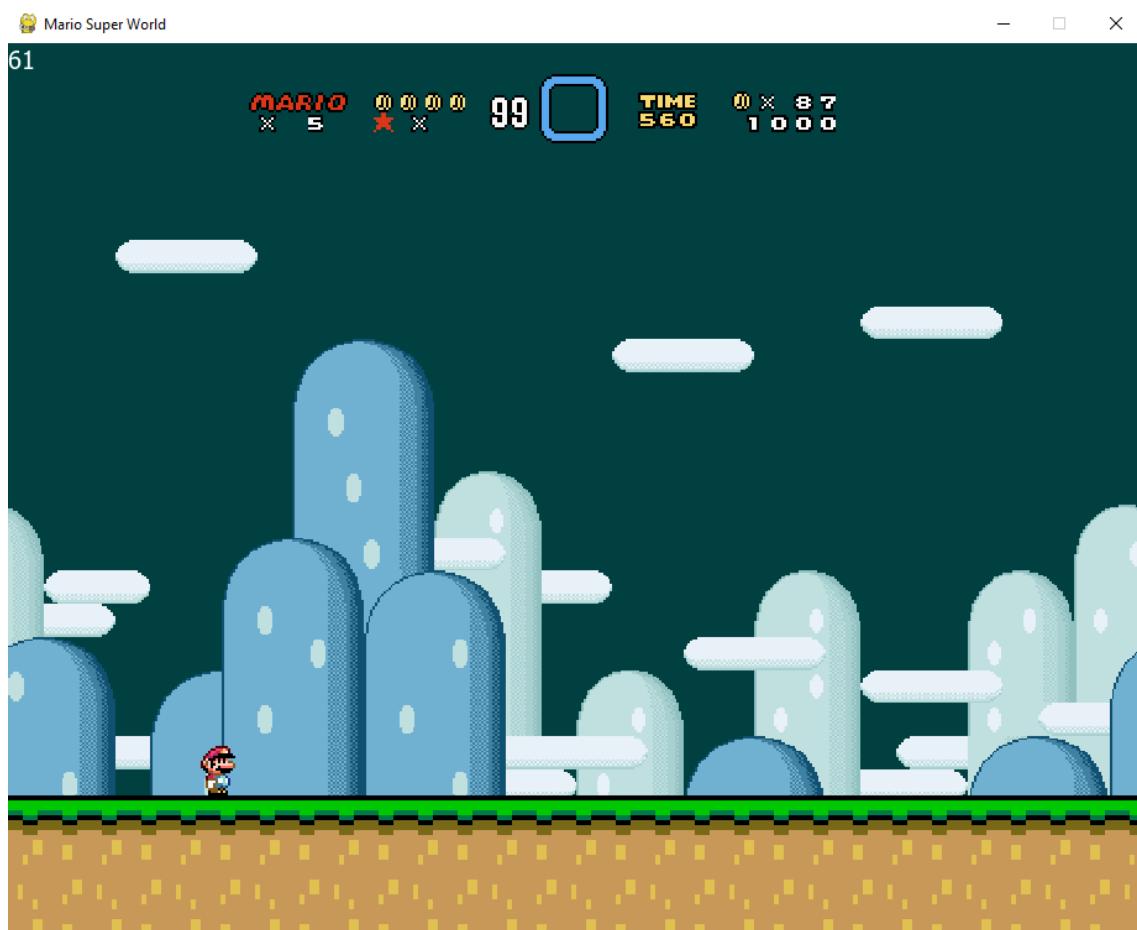


Figura 1 - Protótipo em python

## Planeamento

Esta fase corresponde à escolha da versão do Mario definitiva que ia usar até ao final do projeto, linguagem de programação e também uma lista do que eu preciso para ter um jogo funcional.

Nessa lista continha os seguintes objetivos:

- Movimentação do personagem principal(Mario)
- Colisões entre entidades
- ScreenManager para possibilitar diferentes telas
- Uma classe responsável pelo spawn de entidades e carregamento dos níveis.
- Diferentes powerup como Cogumelo,Flor
- PopUps que dão spawn quando eu mato inimigo para dar feedback do quanto score o jogador ganhou
- Diferentes tipos de inimigos como goomba ,koopa,planta carnívora
- Algo que facilite o carregamento de múltiplas imagens (SpriteSheet)
- Bandeira para ir próximo nível também como a animação do mario a segurar o mastro da bandeira
- Tubos que permitem mudar de cenário.
- Criação de uma scoreboard tanto online como offline

## Aquisição do material

Nesta etapa foi buscar sprite(imagens) do inimigos e spritesheet(conjunto de imagens) para isso contei com ajuda do site: “<https://www.spriters-resource.com/>” onde tem varios sprites,cenarios,fontes de jogos do estilo 2D.

## Preparação do ambiente de trabalho

Começo por instalar visual studio code(editor de texto), depois disso instalei mingw(pasta onde vem compilador gcc e g++) e foi site do SDL2 instalei as bibliotecas SDL(SDL2\_IMAGE,SDL2,SDL2\_MIXER).

## Criação do jogo

Antes mesmo de programar criei um makefile para facilitar na hora da compilação.

Nesta fase onde começa parte da codificação do jogo, começo por criar parte principal de qualquer engine isto Game, Timer, InputManager, Drawings, Sprite, SoundManager.

Depois da parte principal começo por criar diferentes tipos de telas e o GUI (Interface gráfica do utilizador).

E por último faço carregador de níveis, gerenciador de entidades e os diferente tipos de entidades.

Abaixo estarão imagens do código e o resumo do que o código faz:

## EntryPoint

O ponto de entrada a “primeira” função que o programa executa.

Onde iniciamos o jogo e criamos o loop.

```
3/8/22, 5:58 PM                                main.cpp
1 #include "Game.h"
2 int main(int argc, char* argv[])
3 {
4     game.init();
5     game.loop();
6     return 0;
7 }
8
```

*Figura 2 - Função main*

## Makefile

Makefile cria comando para g++(compiler) poder compilar o código tendo quatro tarefas:

**make debug** - cria executável no bin/Debug/ sem otimização -O3 e com debug symbols, permitindo criar breakpoint no código e ver onde acontece as segmentation faults.

**make** - cria executável no bin/Release com otimização -O3 e sem debug symbols sendo mais rápido e mais leve.

**make clean** – Elimina todos os object files para poder dar rebuild a solução

**make dirs** – Util no início do projeto cria a seguinte estrutura de pastas:

src -> main.cpp

bin/Release e bin/Debug

include(onde ficam os headers)

```

3/8/22, 4:57 PM                                         Makefile
1 CC = g++
2 CFLAGS = -IC:/mingw64/include -Wall -Iinclude#include flags
3 LMYSQ = -LC:/mingw64/lib -lmysqlclient -lmysql
4 LDFLAGS = -LC:/mingw64/lib -lmingw32 -lSDL2main -lSDL2_image -lSDL2_mixer -lSDL2_ttf
      -lSDL2 #linker flags
5 SRC = $(wildcard src/*.cpp src/Core/*.cpp src/Screens/*.cpp src/GUI/*.cpp
      src/Entities/*.cpp)
6 OBJ = $(SRC:.cpp=.o)
7
8 BIN = bin/Release/SuperMario.exe
9 all: CFLAGS += -O3
10 all: LDFLAGS += -s
11 all: link
12 all: execute
13
14 debug: CFLAGS += -DDEBUG -g
15 debug: BIN = bin/Debug/SuperMario.exe
16 debug: link
17 debug: execute
18 dirs:
19   mkdir src
20   @echo > src/main.cpp
21   mkdir include
22   mkdir bin
23   cd bin && mkdir Debug && mkdir Release
24 execute:
25   $(BIN)
26
27 link: $(OBJ)
28   $(CC) $(LMYSQ) $^ $(LDFLAGS) -o $(BIN)
29
30 %.o: %.cpp
31   $(CC) -o $@ -c $< $(CFLAGS)
32
33 clean:
34   del src\*.o  src\Core\*.o src\GUI\*.o src\Entities\*.o  src\Screens\*.o
35
36
37

```

*Figura 3 - MakeFile*

## Core Game

Esta classe é responsável por criar janela, renderizar e gerenciar os eventos da tela atual.

**init** – Inicia o SDL2, cria a janela e o renderer.

**loop** – Começa loop do jogo onde limita a 60 fps e chama função events e render

**events** - Função responsável por capturar o pressionamento das teclas e guardar na classe InputManager.

```
3/8/22, 4:59 PM                                         Game.h
1 #include <SDL2/SDL.h>
2 #include <SDL2/SDL_image.h>
3 #include <stdio.h>
4 #include <SDL2/SDL_ttf.h>
5
6 extern SDL_Renderer* renderer;
7 extern SDL_Window* window ;
8
9 class Game
10 {
11     public:
12         SDL_Color backgroundColor = {0, 0, 0, 255};
13         int width, height;
14         bool isRunning = true;
15         bool isPaused = false;
16         bool isFullscreen = false;
17         int scale = 1;
18
19         bool init(); //init SDL and create window
20         void loop(); //main loop
21         void setScale(int scale);
22         void setFullscreen(bool fullscreen);
23         void events();
24         void render();
25     private:
26 };
27
28 extern Game game;
29
30
31
32
33
```

*Figura 4 - Definições da classe Game*

## [Criação do jogo] – Core

```
3/8/22, 5:02 PM                               Game.cpp
1 #include "ScreenManager.h"
2 #include "Game.h"
3 #include "Timer.h"
4 #include <stdio.h>
5 #include <windows.h>
6 #include "CustomFont.h"
7 #include "Hud.h"
8 #include "World.h"
9 #include "Input.h"
10 #include "Particle.h"
11 #include "SoundManager.h"
12
13 #define WINDOW_WITDH 384 // 384 * 5 = 1920
14 #define WINDOW_HEIGHT 224// 216 * 5 = 1080
15 #define GAME_TITLE "Super mario made by : Rodrigo Reis"
16
17 Game game;
18 SDL_Renderer* renderer;
19 SDL_Window* window ;
20
21 bool Game::init()
22 {
23     AllocConsole();
24     freopen("CONOUT$", "w", stdout);
25
26     if(SDL_Init(SDL_INIT_EVERYTHING) != 0)
27     {
28         printf("SDL_Init Error: %s\n", SDL_GetError());
29         return false;
30     }
31
32     window = SDL_CreateWindow(GAME_TITLE, SDL_WINDOWPOS_CENTERED,
33                               SDL_WINDOWPOS_CENTERED, WINDOW_WITDH, WINDOW_HEIGHT, SDL_WINDOW_SHOWN);
33
34
35     if(window == NULL)
36     {
37         printf("SDL_CreateWindow Error: %s\n", SDL_GetError());
38         return false;
39     }
40
41     renderer = SDL_CreateRenderer(window,0, 0);
42     if(renderer == NULL)
43     {
44         printf("SDL_CreateRenderer Error: %s\n", SDL_GetError());
45     }
46     game.witdh = WINDOW_WITDH;
47     game.height = WINDOW_HEIGHT;
48
49     return true;
50 }
51
52
53 void Game::events()
54 {
55     SDL_Event event;
56     inputReset();
57     while(SDL_PollEvent(&event))
58     {
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148

```

## [Criação do jogo] – Core

```
3/8/22, 5:02 PM                               Game.cpp
59     switch(event.type)
60     {
61         case SDL_QUIT:
62             isRunning = false;
63             break;
64         screenManager.getCurrentScreen()->events(event);
65         inputEvents(event);
66     }
67 }
68
69 void Game::render()
70 {
71     SDL_SetRenderDrawColor(renderer, game.backgroundColor.r, game.backgroundColor.g,
72     game.backgroundColor.b, game.backgroundColor.a);
73     SDL_RenderClear(renderer);
74     screenManager.getCurrentScreen()->render();      //Render current screen
75     SDL_RenderPresent(renderer);
76 }
77
78 void Game::loop()
79 {
80     defaultFont = CustomFont("0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ*!.-x& ",
81     "Resources/FONT/HudFont.png", 8, 8);
82     SDL_RenderSetLogicalSize(renderer, WINDOW_WITDH, WINDOW_HEIGHT);
83     setScale(3);
84     screenManager.init();
85     particleSystem.init();
86     soundManager.init();
87     hud.init();
88     world.init();
89     screenManager.getCurrentScreen()->onChangeScene();
90     while (isRunning)
91     {
92         timer.tick(60);
93         events();
94         render();
95         screenManager.updateChanges();
96     }
97     soundManager.destroy();
98 }
99
100 void Game::setScale(int scale)
101 {
102     if(this->isFullscreen)
103         setFullscreen(false);
104
105     SDL_SetWindowSize(window, WINDOW_WITDH * scale, WINDOW_HEIGHT * scale);
106     this->scale = scale;
107     SDL_SetWindowPosition(window, SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED);
108 }
109
110 void Game::setFullscreen(bool fullscreen)
111 {
112     if(fullscreen)
113     {
114         SDL_DisplayMode dm;
115         SDL_GetCurrentDisplayMode(0, &dm);
116         SDL_SetWindowSize(window, dm.w, dm.h);
```

*Figura 6 - Código da classe Game*

```
3/8/22, 5:02 PM                               Game.cpp
117
118     SDL_SetWindowFullscreen(window, SDL_WINDOW_FULLSCREEN);
119     this->isFullscreen = true;
120 }
121 else
122 {
123     SDL_SetWindowFullscreen(window, 0);
124     setScale(this->scale);
125     this->isFullscreen = false;
126 }
```

*Figura 7 - Código da classe Game*

## Timer

Classe responsável por limitar os fps e calcular deltaTime(diferença do tempo entre frame anterior e o próximo importante para frame inpendence).

Frame inpendence é objeto andar mesma velocidade por segundo independentemente dos quadros(frames) por segundo da aplicação.

Ex: Luigi anda 60 pixels por segundo, mas 30 fps anda 2 pixels por frame e 60 fps anda 1 pixel por frame (velocity = 3600 \* timer.deltaTime).

```
3/8/22, 5:08 PM                               Timer.h
1 #pragma once
2
3 class Timer
4 {
5 public:
6     Timer();
7     void tick(int LimitFps); //Update the clock
8     int getFps() {return 1 / deltaTime;} //Return framerate
9
10    //FPS = 60 -> deltaTime = 1/60 = 0.0166666666666667
11    float deltaTime = 0;
12    int startTick = 0;
13 private:
14 };
15
16 extern Timer timer;
17
18 #define ANIMATION(startSpriteIndex,numberSprites,delay) (timer.startTick / delay ) %
numberSprites + startSpriteIndex
19
```

*Figura 8 - Definições da classe Timer*

```
3/8/22, 5:10 PM                               Timer.cpp
1 #include "Timer.h"
2 #include <SDL2/SDL.h>
3 #include <chrono>
4
5 Timer timer = Timer();
6
7 Timer::Timer()
8 {
9     startTick = 0;
10 }
11
12 using namespace std::chrono;
13 typedef high_resolution_clock Clock;
14 Clock::time_point lastTime;
15
16 void Timer::tick(int LimitFps)
17 {
18     Clock::time_point time = Clock::now();
19     deltaTime = duration_cast<nanoseconds>(time - lastTime).count() / 1000000000.0f;
20     lastTime = time;
21
22     Uint32 EndTick = SDL_GetTicks();
23     int tempDt = EndTick - startTick;
24     if ((1000 / LimitFps) > tempDt) {
25         SDL_Delay(1000 / LimitFps - (tempDt));
26     }
27     startTick = SDL_GetTicks();
28 }
```

*Figura 9 - Código da classe Timer*

## Drawings

Neste ficheiro de código estão guardados funções para desenhar figuras geométricas (quadrado preenchido, bordas de um quadrado, círculo preenchido, linha, etc.).

```
3/8/22, 5:15 PM                                         Drawings.h
1 #include <SDL2/SDL.h>
2
3 void drawFilledRect(SDL_Renderer* renderer,int x,int y,int w,int h,SDL_Color color);
4 void drawRect(SDL_Renderer* renderer,int x,int y ,int w ,int h ,int
   thickness,SDL_Color color);
5 void drawLine(SDL_Renderer* renderer,int x1,int y1,int x2,int y2,SDL_Color color);
6 void drawCircle(SDL_Renderer* renderer,int32_t centreX, int32_t centreY, int32_t
   radius);
7 void drawFilledCircle(SDL_Renderer* renderer,int x, int y, int radius,SDL_Color
   color);
```

*Figura 10 - Definição da função do ficheiro Drawings.h*

## [Criação do jogo] – Core

```
3/8/22, 5:17 PM Drawings.cpp
1 #include "Drawings.h"
2
3 void drawFilledRect(SDL_Renderer* renderer ,int x,int y,int w,int h , SDL_Color color)
4 {
5     SDL_Rect rect = {x,y,w,h};
6     SDL_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a);
7     SDL_RenderFillRect(renderer, &rect);
8 }
9
10 void drawRect(SDL_Renderer* renderer,int x,int y ,int w ,int h ,int thickness,SDL_Color color)
11 {
12     drawFilledRect(renderer,x - thickness,y,thickness,h,color); // Left border -> |
13     drawFilledRect(renderer,x + w,y,thickness,h,color); // Right border -> |
14     drawFilledRect(renderer,x-thickness,y- thickness ,w + thickness*2,thickness,color);
15     // Top border -> -
16     drawFilledRect(renderer,x-thickness,y+h,w + thickness*2,thickness,color); // Bottom border -> -
17 }
18 void drawLine(SDL_Renderer* renderer,int x1,int y1,int x2,int y2,SDL_Color color)
19 {
20     SDL_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a);
21     SDL_RenderDrawLine(renderer, x1, y1, x2, y2);
22 }
23
24 //https://stackoverflow.com/questions/38334081/howto-draw-circles-arcs-and-vector-
25 void drawCircle(SDL_Renderer* renderer,int32_t centreX, int32_t centreY, int32_t radius)
26 {
27     const int32_t diameter = (radius * 2);
28
29     int32_t x = (radius - 1);
30     int32_t y = 0;
31     int32_t tx = 1;
32     int32_t ty = 1;
33     int32_t error = (tx - diameter);
34
35     while (x >= y)
36     {
37         // Each of the following renders an octant of the circle
38         SDL_RenderDrawPoint(renderer, centreX + x, centreY - y);
39         SDL_RenderDrawPoint(renderer, centreX + x, centreY + y);
40         SDL_RenderDrawPoint(renderer, centreX - x, centreY - y);
41         SDL_RenderDrawPoint(renderer, centreX - x, centreY + y);
42         SDL_RenderDrawPoint(renderer, centreX + y, centreY - x);
43         SDL_RenderDrawPoint(renderer, centreX + y, centreY + x);
44         SDL_RenderDrawPoint(renderer, centreX - y, centreY - x);
45         SDL_RenderDrawPoint(renderer, centreX - y, centreY + x);
46
47         if (error <= 0)
48         {
49             ++y;
50             error += ty;
51             ty += 2;
52         }
53
54         if (error > 0)
```

*Figura 11 - Código do ficheiro Drawings.h*

## [Criação do jogo] – Core

```
3/8/22, 5:17 PM Drawings.cpp
55     {
56         --x;
57         tx += 2;
58         error += (tx - diameter);
59     }
60 }
61 }
62
63
64 void drawFilledCircle(SDL_Renderer* renderer,int x, int y, int radius,SDL_Color color)
65 {
66     SDL_SetRenderDrawColor(renderer, color.r,color.g,color.b,color.a);
67     for (int w = 0; w < radius * 2; w++)
68     {
69         for (int h = 0; h < radius * 2; h++)
70         {
71             int dx = radius - w; // horizontal offset
72             int dy = radius - h; // vertical offset
73             if ((dx*dx + dy*dy) <= (radius * radius))
74             {
75                 SDL_RenderDrawPoint(renderer, x + dx, y + dy);
76             }
77         }
78     }
79 }
80
81 }
```

*Figura 12 – Código do ficheiro Drawings.cpp*

## Input

Este ficheiro código serve para facilitar/gerir o input.

Ex: if(isKeyDown(SDL\_SCANCODE\_A)){printf("a")}; -> se estiver presionar tecla a da print a na consola.

**inputEvents** – Esta função é chamada na função events da classe Game e guarda o presionamento de teclas em dois array e guarda a posição do mouse.

**inputReset** – É chamada no inicio da função events da classe Game e poem o array KeyPressed a zeros.

**isKeyDown** – retorna verdadeiro se tecla(index código da tecla) estiver ser pressionada.

**isKeyPressed** – retorna verdadeiro se tecla(index código da tecla) se estiver sido solta.

```
3/8/22, 5:36 PM                                Input.h
1 #include "Vec2.h"
2 #include <SDL2/SDL.h>
3
4 void inputEvents(SDL_Event& event);
5 void inputReset();
6
7 bool isKeyDown(int index);
8 bool isKeyPressed(int index);
9 bool isLeftButtonDown();
10 bool isRightButtonDown();
11 Vec2 getMousePos();
12 bool isLeftMouseClicked();
13 bool isRightMouseClicked();
14
```

*Figura 13 - Definições do ficheiro Input.h*

## [Criação do jogo] – Core

```
3/8/22, 5:41 PM                               Input.cpp
1 #include "Input.h"
2 #include <string.h>
3 #include <stdio.h>
4 #include <cassert>
5
6 #define MAX_KEYS 256
7 bool KeysDown[MAX_KEYS];
8 bool KeysPressed[MAX_KEYS];
9 bool leftButtonDown = false;
10 bool rightButtonDown = false;
11 bool leftButtonUp = false, rightButtonUp = false;
12 int mousePosX = 0, mousePosY = 0;
13
14 void inputEvents(SDL_Event& event)
15 {
16     switch (event.type)
17     {
18         case SDL_KEYDOWN:
19             if(event.key.keysym.scancode < MAX_KEYS)
20             {
21                 KeysDown[event.key.keysym.scancode] = true;
22             }
23             break;
24         case SDL_KEYUP:
25             if(event.key.keysym.scancode < MAX_KEYS)
26             {
27                 KeysDown[event.key.keysym.scancode] = false;
28                 KeysPressed[event.key.keysym.scancode] = true;
29             }
30             break;
31         case SDL_MOUSEBUTTONDOWN:
32             if(event.button.button == SDL_BUTTON_LEFT)
33                 leftButtonDown = true;
34             else if(event.button.button == SDL_BUTTON_RIGHT)
35                 rightButtonDown = true;
36             break;
37         case SDL_MOUSEBUTTONUP:
38             if(event.button.button == SDL_BUTTON_LEFT)
39             {
40                 leftButtonDown = false;
41                 leftButtonUp = true;
42             }
43             else if(event.button.button == SDL_BUTTON_RIGHT)
44             {
45                 rightButtonDown = false;
46                 rightButtonUp = true;
47             }
48             break;
49         case SDL_MOUSEMOTION:
50             mousePosX = event.motion.x;
51             mousePosY = event.motion.y;
52             break;
53
54     }
55 }
56
57
58
59 void inputReset()
```

**Figura 14 - Código do Input**

```
3/8/22, 5:41 PM                               Input.cpp
60 {
61     leftButtonUp = false;
62     rightButtonUp = false;
63     memset(KeysPressed, 0, sizeof(KeysPressed));
64 }
65
66 bool isKeyDown(int index) { return KeysDown[index]; }
67 bool isKeyPressed(int index) { return KeysPressed[index]; }
68 bool isLeftButtonDown() { return leftButtonDown; }
69 bool isRightButtonDown() { return rightButtonDown; }
70 Vec2 getMousePos() { return Vec2(mousePosX, mousePosY); }
71 bool isLeftMouseClicked() { return leftButtonUp; }
72 bool isRightMouseClicked() { return rightButtonUp; }
73
```

**Figura 15 - Código do Input**

## SoundManager

SoundManager ajuda me tocar musicas , SFX(efeitos sonoros) e reduzir ou aumentar o volumes da musicas ou efeitos sonoros usando biblioteca SDL\_MIXER.

```
3/8/22, 5:47 PM                               SoundManager.h
1 #pragma once
2 #include <vector>
3 #include <string>
4 #include <SDL2/SDL_mixer.h>
5
6 #define MAX_VOLUME 30
7 class SoundManager
8 {
9     public:
10     void init();
11     Mix_Music* loadMusic(std::string musicName);
12     void playMusic(std::string musicName,int loops = 1);
13     void playSFX(std::string soundName,int loops = 0);
14
15     void stopMusic();
16     void stopMusic(Mix_Music* music);
17     void destroy(Mix_Music* music);
18     void destroy();
19
20     void setMusicVolume(int volume);
21     void setSFXVolume(int volume);
22     bool isPlayingMusic = false;
23     bool isPlayingSFX = false;
24
25     int musicVolume = 1;
26     int sfxVolume = 1;
27     std::string currentMusicName = "";
28     private:
29     Mix_Chunk* currentSFX = nullptr;
30     Mix_Music* currentMusic = nullptr;
31 };
32
33 extern SoundManager soundManager;
```

*Figura 16 - Definições SoundManager*

## [Criação do jogo] – Core

```
3/8/22, 5:54 PM                                     SoundManager.cpp
1 #include <stdio.h>
2 #include "SoundManager.h"
3
4 #define MUSIC_PATH "Resources/Sounds/Music/"
5 #define SFX_PATH "Resources/Sounds/SFX/"
6
7 SoundManager soundManager;
8
9 void finnishedMusic(){printf("Music finished\n");}
10
11 void SoundManager::init()
12 {
13     if(Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048) < 0)
14     {
15         printf("SDL_mixer Error: %s\n", Mix_GetError());
16         return;
17     }
18     Mix_HookMusicFinished(finnishedMusic);
19     this->setMusicVolume(this->musicVolume);
20     this->setSFXVolume(this->sfxVolume);
21 }
22
23 void SoundManager::playMusic(std::string musicName,int loops)
24 {
25     Mix_FreeMusic(currentMusic);
26     std::string musicPath = MUSIC_PATH + musicName;
27     currentMusic = Mix_LoadMUS(musicPath.c_str());
28     currentMusicName = musicName;
29     Mix_PlayMusic(currentMusic,loops);
30 }
31
32 void SoundManager::playSFX(std::string soundName,int loops)
33 {
34     Mix_FreeChunk(currentSFX);
35     std::string sfxPath = SFX_PATH + soundName;
36     currentSFX = Mix_LoadWAV(sfxPath.c_str());
37     if(currentSFX == nullptr){ printf("Failed to load sfx! SDL_mixer Error: %s\n",
38     Mix_GetError()); }
39     Mix_PlayChannel(0, currentSFX, loops);
40 }
41 void SoundManager::stopMusic(){Mix_HaltMusic();}
42
43 void SoundManager::setMusicVolume(int volume)
44 {
45     this->musicVolume = volume;
46
47     if(musicVolume > MAX_VOLUME){ this->musicVolume = MAX_VOLUME; }
48     else if(musicVolume < 0){musicVolume = 0;}
49
50     Mix_VolumeMusic(musicVolume * 4.26F);
51 }
52
53 void SoundManager::setSFXVolume(int volume)
54 {
55     this->sfxVolume = volume;
56
57     if(sfxVolume > MAX_VOLUME){ this->sfxVolume = MAX_VOLUME; }
```

**Figura 17 - Código do classe SoundManager**

```
3/8/22, 5:54 PM                                     SoundManager.cpp
59
60     Mix_Volume(0,sfxVolume * 4.26F);
61 }
62
63 //free the memory
64 void SoundManager::destroy(){
65     Mix_FreeMusic(currentMusic);
66     Mix_FreeChunk(currentSFX);
67     Mix_CloseAudio();
68 }
```

**Figura 18 - Código da classe SoundManager**

## Sprite

Classe responsável por carregar imagens, desenhar imagens na tela e tirar fundo das imagens.

Também tem funções adicionais como carregar SpriteSheet(atlas de textura onde tem várias imagens menores dentro de uma imagem, geralmente agrupadas) e carregar todas imagens dentro de uma pasta.

```
3/8/22, 6:24 PM                               Sprite.h

1 #pragma once
2 #include <SDL2/SDL.h>
3 #include <vector>
4 #include <string>
5
6 class Sprite
7 {
8     public:
9         ~Sprite();
10        Sprite(){}
11        void draw(int x,int y,SDL_RendererFlip flip = SDL_FLIP_NONE);
12        Sprite(const char* filename);
13        Sprite(const char* filename,SDL_Color transparentColor);
14        Sprite(SDL_Surface* image_surface, int w, int h,SDL_Color transparentColor);
15        void destroy();
16    private:
17    public:
18        SDL_Texture* texture = NULL;
19        int w = 0,h = 0;
20    };
21
22 struct SpreadSheetInfo
23 {
24     int width, height;
25     SDL_Color transparent;
26     int margin;
27 };
28
29 void loadSpriteSheet(std::vector<Sprite>& sprites,const char* imagePath);
30 void loadMultipleImages(std::vector<Sprite>& sprites, std::string FolderPath,
   std::string Base_imageName); //load all textures in folder
31
32
```

**Figura 19 - Definições da classe Sprite**

## [Criação do jogo] – Core

```
3/8/22, 6:28 PM                               Sprite.cpp
1 #include "Sprite.h"
2 #include <SDL2/SDL_image.h>
3 #include "Game.h"
4 #include "Utils.h"
5 #include <stdio.h>
6 #include <cassert>
7 Sprite::~Sprite(){}
8
9 Sprite::Sprite(const char* filename)
10 {
11     if(!fileExists(filename))
12     {
13         printf("File not found: %s\n",filename);
14         return;
15     }
16
17     SDL_Surface* imageSurface = IMG_Load(filename);
18     this->w = imageSurface->w;
19     this->h = imageSurface->h;
20     this->texture = SDL_CreateTextureFromSurface(renderer, imageSurface);
21     SDL_FreeSurface(imageSurface);
22 }
23
24
25 Sprite::Sprite(const char* filename,SDL_Color transparentColor)
26 {
27     if(!fileExists(filename))
28     {
29         printf("File not found: %s\n",filename);
30         return;
31     }
32
33     SDL_Surface* imageSurface = IMG_Load(filename);
34     this->w = imageSurface->w;
35     this->h = imageSurface->h;
36     Uint32 colorkey = SDL_MapRGB(imageSurface->format, transparentColor.r,
37     transparentColor.g, transparentColor.b);
38     SDL_SetColorKey(imageSurface, SDL_TRUE, colorkey);
39     this->texture = SDL_CreateTextureFromSurface(renderer, imageSurface);
40     SDL_FreeSurface(imageSurface);
41 }
42
43 Sprite::Sprite(SDL_Surface* image_surface, int w, int h,SDL_Color transparentColor)
44 {
45     this->w = w;
46     this->h = h;
47     Uint32 colorkey = SDL_MapRGB(image_surface->format, transparentColor.r,
48     transparentColor.g, transparentColor.b);
49     SDL_SetColorKey(image_surface, SDL_TRUE, colorkey);
50     this->texture = SDL_CreateTextureFromSurface(renderer, image_surface);
51     SDL_FreeSurface(image_surface);
52 }
53
54 void Sprite::draw(int x,int y,SDL_RendererFlip flip)
55 {
56     if(this->w == 0 || this->h == 0) return;
57     assert(this->texture != nullptr);
```

*Figura 20 - Código da classe Sprite*

## [Criação do jogo] – Core

```
3/8/22, 6:28 PM                               Sprite.cpp
58     SDL_Point center= SDL_Point();
59     SDL_RenderCopyEx(renderer, this->texture, NULL, &rect, 0, &center, flip);
60 }
61
62 //get image in certain cell of sprite sheet
63 Sprite getImage(SDL_Surface* sheet, int FrameX, int FrameY, int margin, int width,
64 int height, SDL_Color transparent)
65 {
66     int x = FrameX * width + margin * FrameX;
67     int y = FrameY * height + margin * FrameY;
68
69     SDL_Surface* image = SDL_CreateRGBSurface(0, width, height, 32, 0, 0, 0, 0);
70
71     SDL_Rect src = { x,y,width,height };
72     SDL_Rect dest = { 0,0,width,height };
73     SDL_BlitSurface(sheet, &src, image, &dest);
74
75     return Sprite(image, width, height, transparent);
76 }
77 void loadSpriteSheet(std::vector<Sprite>& sprites,const char* imagePath)
78 {
79     std::string txtFile = swapExtension(imagePath, "txt");
80
81     //if file not found assert
82     if(!fileExists(txtFile.c_str()))
83     {
84         printf("imagePath: %s\n",txtFile.c_str());
85         system("pause");
86         assert(( false && "Sprite text configuration file not found"));
87         exit(1);
88     }
89
90     FILE* file = fopen(txtFile.c_str(),"r");
91     SpreadSheetInfo info;
92     char buffer[256];
93     fscanf(file,"%s = %i\n",buffer,&info.width);
94     fscanf(file,"%s = %i\n",buffer,&info.height);
95     fscanf(file,"%s =
%hhu,%hhu,%hhu\n",buffer,&info.transparent.r,&info.transparent.g,&info.transparent.b)
96     ;
97     fscanf(file,"%s = %i\n",buffer,&info.margin);
98     fclose(file);
99
100    SDL_Surface* sheet = IMG_Load(imagePath);
101    int totalFrameX = 0;
102    int tempX = 0;
103    while(tempX < sheet->w)
104    {
105        tempX += info.width + info.margin;
106        totalFrameX++;
107    }
108
109    int totalFrameY = sheet->h / info.height;
110
111
112    for(int y = 0;y < totalFrameY;y++)
113    {
114
```

*Figura 21 - Código da classe Sprite*

## [Criação do jogo] – Core

```
3/8/22, 6:28 PM                               Sprite.cpp
115     for(int x = 0;x < totalFrameX;x++)
116     {
117         Sprite sprite = getImage(sheet, x, y, info.margin, info.width, info.height,
118         {info.transparent.r,info.transparent.g,info.transparent.b,255});
119         sprites.push_back(sprite);
120     }
121
122     //free sheet
123     SDL_FreeSurface(sheet);
124 }
125
126 void loadMultipleImages(std::vector<Sprite>& sprites, std::string FolderPath,
127 std::string Base_ImageViewName)
128 {
129     int index = 0;
130     while (true)
131     {
132         std::string imageNames = FolderPath + Base_ImageViewName + std::to_string(index) +
133         ".png";
134         std::string extension = "";
135
136         if(!fileExists(imageNames.c_str())) { break; }
137         if(!getFileExtension(imageNames,extension)) { break;} //if no extension break
138         if(strcmp(extension.c_str(),"png") != 0) { break;} //if extension is not png
139         break;
140
141         std::string txtFile = swapExtension(imageNames, "txt");
142
143         if(fileExists(txtFile.c_str())){
144             loadSpriteSheet(sprites,imageNames.c_str());
145         }
146         else{
147             sprites.push_back(Sprite(imageNames.c_str(),{0, 108, 248,255}));
148         }
149     }
150
151 void Sprite::destroy()
152 {
153     if(texture != nullptr)
154     {
155         SDL_DestroyTexture(texture);
156         texture = nullptr;
157     }
158 }
```

Figura 22 - Código da classe Sprite

## Screens

### ScreenManager

Esta classe guarda diferentes telas(game screen,start menu,options) e executa as funções events e render da janela atual

```
3/8/22, 6:46 PM                               ScreenManager.h
1 #pragma once
2 #include <vector>
3 #include "BaseScreen.h"
4 #include "Sprite.h"
5
6 enum EScreen
7 {
8     SCREEN_NONE = -1,
9     TITLE_SCREEN,
10    OPTIONS_SCREEN,
11    NEW_GAME_SCREEN,
12    NEXT_LEVEL_SCREEN,
13    SCOREBOARD_SCREEN,
14    GAME_SCREEN,
15    GAME_OVER_SCREEN,
16    SCOREBOARD_SELECTION_SCREEN,
17    SUBMIT_SCORE_SCREEN,
18 };
19
20 class ScreenManager
21 {
22 public:
23     ScreenManager();
24     ~ScreenManager();
25     void init();
26     void changeScreen(EScreen index);
27     void updateChanges();
28     BaseScreen* getCurrentScreen();
29     std::vector<Sprite> sprites;
30 private:
31     std::vector<BaseScreen*> screens;
32     EScreen nextScreen = EScreen::SCREEN_NONE;
33     EScreen currentScreen = TITLE_SCREEN;
34 };
35 };
36
37 extern ScreenManager screenManager;
```

*Figura 23 - Definições do SoundManager*

## [Criação do jogo] – Screens

```
3/8/22, 6:50 PM                                         ScreenManager.cpp
 1 #include "ScreenManager.h"
 2 #include <Windows.h>
 3
 4 //SCREENS
 5 #include "TitleScreen.h"
 6 #include "OptionsScreen.h"
 7 #include "NextLevelScreen.h"
 8 #include "ScoreBoardScreen.h"
 9 #include "GameScreen.h"
10 #include "NewGameScreen.h"
11 #include "GameOverScreen.h"
12 #include "ScoreBoardSelection.h"
13 #include "SubmitScoreScreen.h"
14
15 ScreenManager screenManager;
16
17 ScreenManager::ScreenManager()
18 {
19     screens.push_back(new TitleScreen());
20     screens.push_back(new OptionsScreen());
21     screens.push_back(new NewGameScreen());
22     screens.push_back(new NextLevelScreen());
23     screens.push_back(new ScoreBoardScreen());
24     screens.push_back(new GameScreen());
25     screens.push_back(new GameOverScreen());
26     screens.push_back(new ScoreBoardSelection());
27     screens.push_back(new SubmitScoreScreen());
28 }
29
30 ScreenManager::~ScreenManager(){}
31
32 void ScreenManager::changeScreen(EScreen index){nextScreen = index;}
33 void ScreenManager::init(){
34     loadMultipleImages(sprites,"Resources/Screens/TitleScreen/","StartMenu_");
35 }
36
37 void ScreenManager::updateChanges()
38 {
39     if(nextScreen != SCREEN_NONE)
40     {
41         currentScreen = nextScreen;
42         nextScreen = SCREEN_NONE;
43         screens[(int)(currentScreen)]->onChangeScene();
44     }
45 }
46
47 BaseScreen* ScreenManager::getCurrentScreen(){return screens[(int)(currentScreen)];}
48
```

**Figura 24 - Código da classe ScreenManager**

## BaseScreen

Esta classe é herdada por todas telas.

```
3/8/22, 7:40 PM                                BaseScreen.h
1 #pragma once
2 #include <SDL2/SDL.h>
3
4 class BaseScreen
5 {
6 public:
7     //Function responsible for handling events
8     virtual void events(SDL_Event events) {}
9
10    //Function responsible for drawing on the screen
11    virtual void render() {}
12
13    //This function is called when changes from current to this
14    virtual void onChangeScene(){}
15 private:
16 };
17
```

*Figura 25 - Definições da classe BaseScreen*

### TitleScreen

A tela inicial do jogo que tem opção de jogar com um ou dois jogadores, ir para menu de opções e a ver scoreboard.



Figura 26 - Imagem da tela inicial

3/8/22, 10:11 PM TitleScreen.h

```
1 #pragma once
2 #include "BaseScreen.h"
3 #include "Sprite.h"
4
5 class TitleScreen : public BaseScreen
6 {
7 public:
8     void events(SDL_Event events);
9     void render();
10    void onChangeScene();
11 private:
12 };
13
```

Figura 27 - Definições da classe TitleScreen

## [Criação do jogo] – Screens

```
3/8/22, 10:13 PM TitleScreen.cpp
1 #include "TitleScreen.h"
2 #include "Game.h"
3 #include "KeyGui.h"
4 #include "ScreenManager.h"
5 #include "LevelManager.h"
6 #include <Windows.h>
7
8 #include "World.h"
9
10 void TitleScreen::events(SDL_Event events)
11 {
12     keyGui.events(events);
13 }
14 void TitleScreen::render()
15 {
16     world.render();
17     screenManager.sprites[0].draw(110,15);
18     keyGui.begin(150, 125);
19
20     if(keyGui.button("1 PLAYER GAME")) { world.maxPlayers = 1;
screenManager.changeScreen(NEW_GAME_SCREEN); }
21     if(keyGui.button("2 PLAYER GAME")) { world.maxPlayers = 2;
screenManager.changeScreen(NEW_GAME_SCREEN);}
22     if(keyGui.button("SCOREBOARD"))
{
screenManager.changeScreen(SCOREBOARD_SELECTION_SCREEN); }
23     if(keyGui.button("OPTIONS")) { screenManager.changeScreen(OPTIONS_SCREEN); }
24
25     keyGui.end();
26 }
27 void TitleScreen::onChangeScene()
28 {
29     std::string titleLevelPath = "Resources/Levels/TitleScreen.txt";
30     if(titleLevelPath != levelManager.levelPath)
31     {
32         world.numPlayers = 0;
33         world.maxPlayers = 0;
34         levelManager.loadLevel("Resources/Levels/TitleScreen.txt");
35     }
36     keyGui = KeyGui(&defaultFont,& screenManager.sprites[1],0,20);
37 }
```

*Figura 28 - Código da TitleScreen*

## OptionsScreen

Neste menu da para mudar tamanho da janela, alterar o volume da musica e do SFX



Figura 29 - Imagem da tela de opções

```
3/8/22, 10:34 PM OptionsScreen.h
1 #include "BaseScreen.h"
2 #include "Sprite.h"
3
4 class OptionsScreen : public BaseScreen
5 {
6 public:
7     void events(SDL_Event events);
8     void render();
9     void onChangeScene(){}
10 private:
11 };
12
```

Figura 30 - Definições da classe OptionsScreen

## [Criação do jogo] – Screens

```
3/8/22, 10:34 PM OptionsScreen.cpp
1 #include "OptionsScreen.h"
2 #include "Game.h"
3 #include "SoundManager.h"
4 #include "KeyGui.h"
5 #include "ScreenManager.h"
6 #include "World.h"
7 void OptionsScreen::events(SDL_Event events)
8 {
9     keyGui.events(events);
10    if (events.type == SDL_KEYUP && events.key.keysym.sym == SDLK_ESCAPE)
11        {screenManager.changeScreen(TITLE_SCREEN);}
12 }
13 void OptionsScreen::render()
14 {
15     world.render();
16     screenManager.sprites[0].draw(110, 15);
17     keyGui.begin(150, 125);
18     if (keyGui.toggle("FULLSCREEN", &game.isFullscreen)) {
19         game.setFullscreen(game.isFullscreen);
20         if (keyGui.numericInt("MUSIC",&soundManager.musicVolume, 0, MAX_VOLUME)) {
21             soundManager.setMusicVolume(soundManager.musicVolume); }
22             if(keyGui.numericInt("SFX", &soundManager.sfxVolume,0,MAX_VOLUME)) {
23                 soundManager.setSFXVolume(soundManager.sfxVolume); }
24             if (keyGui.numericInt("SCALE",&game.scale, 1, 5))
25                 game.setScale(game.scale);
26             keyGui.end();
27 }
```

*Figura 31 - Código da classe OptionsScreen*

### ScoreBoardSelection

Depois que damos enter no Scoreboard aparece menu a perguntar se queremos aceder ao score offline ou online.

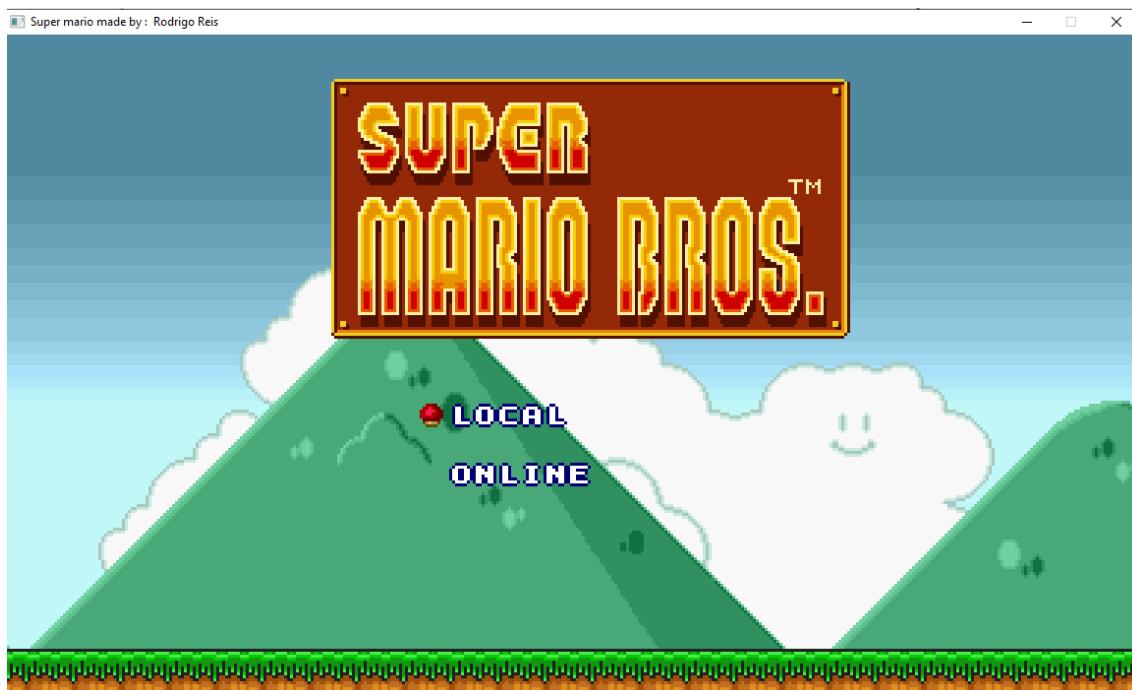


Figura 32 - Imagem da tela de seleção de scoreboard

3/8/22, 11:03 PM

ScoreBoardSelection.h

```
1 #include "BaseScreen.h"
2
3 class ScoreBoardSelection : public BaseScreen
4 {
5 public:
6     void events(SDL_Event events);
7     void render();
8     void onChangeScene();
9 private:
10};
```

Figura 33 - Definições da classe ScoreBoardSelection

## [Criação do jogo] – Screens

```
3/8/22, 11:03 PM ScoreBoardSelection.cpp
1 #include "ScoreBoardSelection.h"
2 #include "ScoreBoardScreen.h"
3 #include "World.h"
4 #include "KeyGui.h"
5 #include "ScreenManager.h"
6 #include "LevelManager.h"
7
8 void ScoreBoardSelection::events(SDL_Event events)
9 {
10     keyGui.events(events);
11     if (events.type == SDL_KEYUP && events.key.keysym.sym == SDLK_ESCAPE)
12     {
13         screenManager.changeScreen(TITLE_SCREEN);
14         world.maxPlayers = 0;
15     }
16 }
17
18 void ScoreBoardSelection::render()
19 {
20     world.render();
21     screenManager.sprites[0].draw(110, 15);
22     keyGui.begin(150, 125);
23
24     if(keyGui.button("LOCAL")){
25         isScoreBoardOnline = false;
26         screenManager.changeScreen(SCOREBOARD_SCREEN);
27     }
28
29     if(keyGui.button("ONLINE")){
30         isScoreBoardOnline = true;
31         screenManager.changeScreen(SCOREBOARD_SCREEN);
32     }
33     keyGui.end();
34 }
35 void ScoreBoardSelection::onChangeScene()
36 {
37     keyGui.currentIndex = 0;
38     std::string titleLevelPath = "Resources/Levels/TitleScreen.txt";
39     if(titleLevelPath != levelManager.levelPath)
40     {
41         levelManager.loadLevel(titleLevelPath);
42     }
43 }
44 }
```

*Figura 34 - Código da tela ScoreBoard Selection*

## Scoreboard

Mostra score dos jogadores, se escolhemos offline e ele ira ler ficheiro binário que esta guardados no documentos mas se for online ira resgatar os scores a base dados mariadb.

Em seguida ordena pelo score de forma decrescente de cima para baixo e adiciona a tabela.

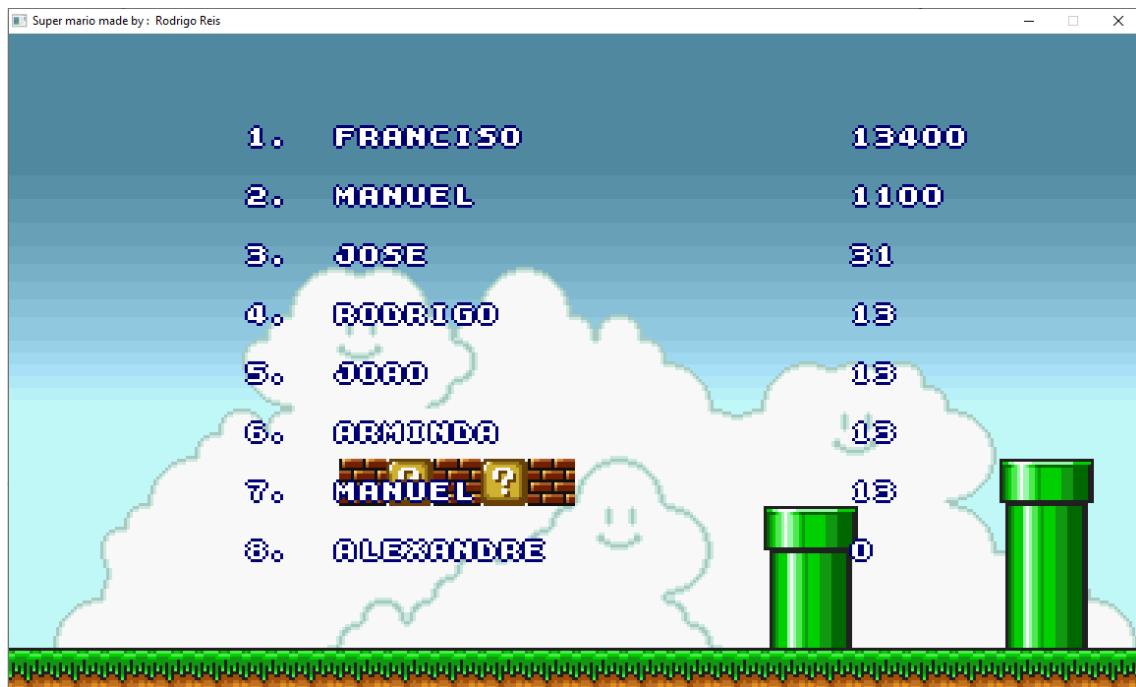


Figura 35 - Imagem da tela de pontuação

```
09/03/22, 14:49                               ScoreBoardScreen.h
1 #pragma once
2 #include "BaseScreen.h"
3 #include "Sprite.h"
4 #include "Table.h"
5
6
7 class ScoreBoardScreen : public BaseScreen
8 {
9 public:
10
11     void events(SDL_Event events);
12     void render();
13     void onChangeScene();
14 private:
15 };
16
17 extern bool isScoreBoardOnline;
```

Figura 36 - Definições da classe ScoreBoard

## [Criação do jogo] – Screens

```
09/03/22, 14:49 ScoreBoardScreen.cpp
1 #include "ScoreBoardScreen.h"
2 #include "ScreenManager.h"
3 #include "LevelManager.h"
4 #include "World.h"
5 #include "KeyGui.h"
6 #include "ScoreSave.h"
7 #include <Windows.h>
8
9 bool isScoreBoardOnline = false;
10 Table scoreboard;
11
12 void ScoreBoardScreen::events(SDL_Event events)
13 {
14     keyGui.events(events);
15
16     if (events.type == SDL_KEYUP && events.key.keysym.sym == SDLK_ESCAPE)
17     {
18         screenManager.changeScreen(SCOREBOARD_SELECTION_SCREEN);
19     }
20 }
21
22 void ScoreBoardScreen::render()
23 {
24     world.render();
25     scoreboard.draw();
26 }
27
28 void ScoreBoardScreen::onChangeScene()
29 {
30     keyGui.currentIndex = 0;
31     levelManager.loadLevel("Resources/Levels/ScoreScreen.txt");
32     world.maxPlayers = -1;
33
34     printf("online: %d\n", isScoreBoardOnline);
35     scoreboard = Table();
36     scoreboard.drawHeader = false;
37     scoreboard.x = 75;
38     scoreboard.y = 25;
39     scoreboard.addCollum("", 30); //Position
40     scoreboard.addCollum("", 175); //Nome
41     scoreboard.addCollum("", 58); //Score
42
43     std::vector<ScoreSaveData> scores;
44     if(isScoreBoardOnline)
45     {
46         if(!scoreSave.loadOnline(scores)){
47             return;
48         }
49     }
50     else{
51         scores = scoreSave.loadLocal();
52     }
53
54     for (int i = 0; i < scores.size(); i++)
55     {
56         if(scores[i].score == -1){break;}
--
```

*Figura 37 - Código da classe ScoreBoard*

```
09/03/22, 14:49 ScoreBoardScreen.cpp
60     scoreboard.addItem(std::to_string(scores[i].score));
61 }
62 }
```

*Figura 38 - Código da classe ScoreBoard*

### NewGameScreen

Se selecionarmos a opção 1PlayerGame ou 2PlayerGame ira para este menu.

Este menu serve para selecionar o save, isto é, podemos continuar o jogo anterior selecionando o save correspondente.



Figura 39 - Imagem da tela NewGameScreen

```
09/03/22, 15:33                                         NewGameScreen.h
1 #include "BaseScreen.h"
2 #include "Sprite.h"
3 #include <string>
4
5 class NewGameScreen : public BaseScreen
6 {
7 public:
8     #define MAX_SLOTS 3
9     void events(SDL_Event events);
10    void render();
11    void onChangeScene();
12 private:
13     std::string slotText[MAX_SLOTS];
14     bool isSlotEmpty[MAX_SLOTS];
15 };
16
```

Figura 40 - Definições da classe NewGameScreen

## [Criação do jogo] – Screens

```
09/03/22, 15:31           NewGameScreen.cpp

1 #include "NewGameScreen.h"
2 #include "World.h"
3 #include "KeyGui.h"
4 #include <sstream>
5 #include "ScreenManager.h"
6 #include "SaveSystem.h"
7 #include "LevelManager.h"
8 #include "Utils.h"
9 #include "Hud.h"
10
11 void NewGameScreen::events(SDL_Event events)
12 {
13     keyGui.events(events);
14
15     if (events.type == SDL_KEYUP && events.key.keysym.sym == SDLK_ESCAPE)
16     {
17         screenManager.changeScreen(TITLE_SCREEN);
18         world.maxPlayers = 0;
19     }
20 }
21
22 void NewGameScreen::render()
23 {
24     world.render();
25
26     screenManager.sprites[0].draw(110, 15);
27     keyGui.begin(150, 125);
28
29     for(int i = 0; i < 3;i++)
30     {
31         if(keyGui.button(slotText[i]))
32         {
33             if(isSlotEmpty[i])
34             {
35                 saveSystem.save(i,{0,0,5,0});
36                 hud.lifes = 5;
37                 hud.setScore(0);
38                 levelManager.currentLevel = 0;
39             }
40             else
41             {
42                 SaveData data;
43                 saveSystem.load(i, data);
44                 levelManager.currentLevel = data.CurrentLevel;
45                 hud.lifes = data.Lives;
46                 hud.setCoin(data.CurrentCoins);
47                 hud.setScore(data.Score);
48             }
49             saveSystem.currentSaveSlot = i;
50             screenManager.changeScreen(NEXT_LEVEL_SCREEN);
51         }
52     }
53
54     keyGui.end();
55
56 }
```

Figura 41 - Código da classe NewGameScreen

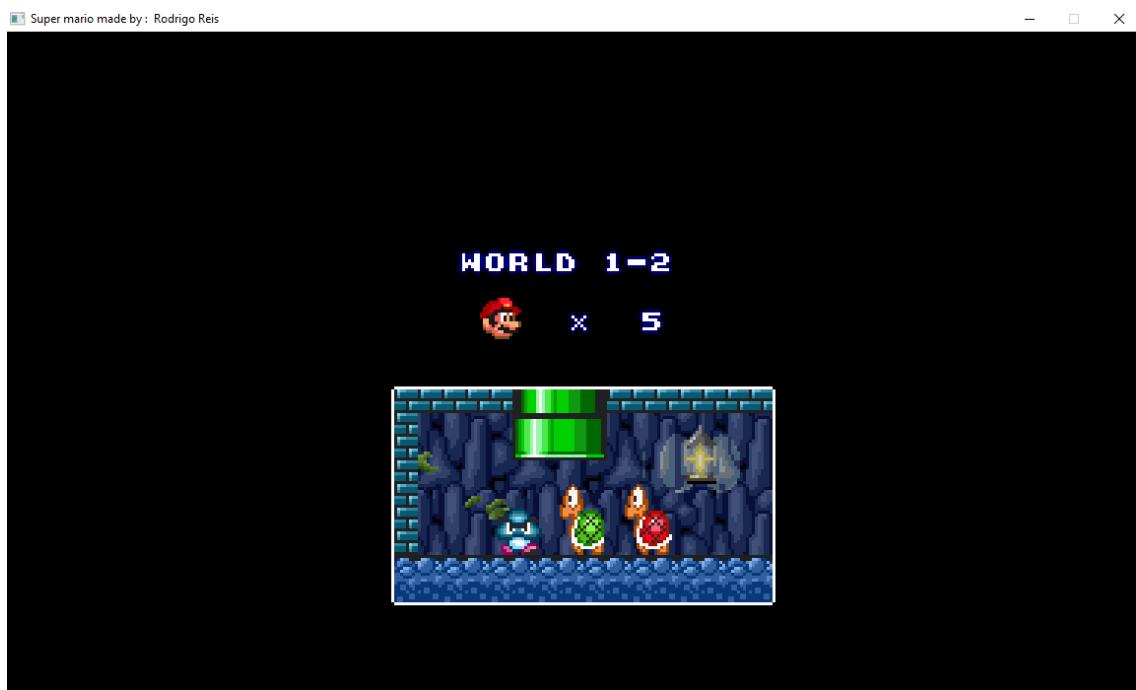
## [Criação do jogo] – Screens

```
09/03/22, 15:31                               NewGameScreen.cpp
60 {
61     keyGui.currentIndex = 0;
62     SaveData saveData;
63     for(int index = 0; index < MAX_SLOTS; index++)
64     {
65         std::string savePath = saveSystem.getPath(index);
66         printf("%s\n", savePath.c_str());
67         if(!fileExists(savePath.c_str()))
68         {
69             isSlotEmpty[index] = true;
70             slotText[index] = "SLOT " + std::to_string(index) + " ... EMPTY";
71             continue;
72         }
73         isSlotEmpty[index] = false;
74         saveSystem.load(index, saveData);
75         slotText[index] = "SLOT " + std::to_string(index) + " LEVEL " +
76         std::to_string(saveData.CurrentLevel+1);
77     }
78 }
```

*Figura 42 - Código da classe NewGameScreen*

### NextLevelScreen

Tela de transição entre NewGameScreen e GameScreen.



3/10/22, 8:35 PM

NextLevelScreen.h

```
1 #pragma once
2 #include "BaseScreen.h"
3 #include "Sprite.h"
4
5 class NextLevelScreen : public BaseScreen
6 {
7 public:
8     ~NextLevelScreen();
9     void events(SDL_Event events);
10    void render();
11    void onChangeScene();
12 private:
13     #define DELAY_NEXT_LEVEL 1000
14     int nextLevelTime = 0;
15     std::string displayName = "";
16     std::vector<Sprite> sprites;
17 };
18
```

*Figura 43 - Definições da classe NextLevelScreen*

## [Criação do jogo] – Screens

```
3/10/22, 8:36 PM                                         NextLevelScreen.cpp

1 #include "NextLevelScreen.h"
2 #include "ScreenManager.h"
3 #include "Game.h"
4 #include "LevelManager.h"
5 #include "CustomFont.h"
6 #include <sstream>
7 #include "World.h"
8 #include "Hud.h"
9 #include "Logger.h"
10
11
12
13 #include "Sprite.h"
14 NextLevelScreen::~NextLevelScreen()
15 {
16     log("NextLevelScreen destructor called\n");
17 }
18
19 void NextLevelScreen::events(SDL_Event events){}
20 void NextLevelScreen::render()
21 {
22     defaultFont.draw(153,74,displayName);
23     std::stringstream ss; ss << "x " << hud.lives;
24     defaultFont.draw(190,94,ss.str());
25     if(world.maxPlayers == 1) { sprites[1].draw(160,90); }
26     else if(world.maxPlayers == 2) {
27         sprites[1].draw(150,90);
28         sprites[2].draw(168,89);
29     }
30     sprites[0].draw(130,120);
31
32     if(SDL_GetTicks() > nextLevelTime + DELAY_NEXT_LEVEL) {
33         screenManager.changeScreen(GAME_SCREEN);
34     }
35
36 }
37 void NextLevelScreen::onChangeScene()
38 {
39     Mix_HaltMusic();
40     displayName = "WORLD ";
41     nextLevelTime = SDL_GetTicks();
42
43     //loop through all sprites and destroy texture
44     for(int i = 0; i < sprites.size(); i++)
45     {
46         SDL_DestroyTexture(sprites[i].texture);
47     }
48     sprites.clear();
49
50     char levelFrameName[100];
51     sprintf(levelFrameName, "Resources/Levels/Level%d.png",
levelManager.currentLevel+1);
52
53     char levelConfig[100];
54     sprintf(levelConfig, "Resources/Levels/Level%d.cfg", levelManager.currentLevel+1);
55     FILE *file = fopen(levelConfig, "r");
56     if(file == NULL) {
57         log("Error opening level config file\n");
58         return;
```

*Figura 44 - Código da classe NextLevelScreen*

```
3/10/22, 8:36 PM                                         NextLevelScreen.cpp

59 }
60 else{
61     char buffer[100];
62     char name[32];
63     int maxtime = 0;
64     fscanf(file, "%s",name);
65     fscanf(file, "%f", &hud.countdownTimerMax);
66     levelManager.displayName = name;
67     hud.countdownTimer = hud.countdownTimerMax;
68     displayName += name;
69 }
fclose(file);
71
72
73 char marioIconName[] = "Resources/Controller/Mario/icon.png";
74 char luigiIconName[] = "Resources/Controller/Luigi/icon.png";
75 sprites.push_back( Sprite(levelFrameName));
76 sprites.push_back( Sprite(marioIconName,{0,108,248}));
77 sprites.push_back( Sprite(luigiIconName,{0,108,248}));
```

*Figura 45 - Código da classe NextLevelScreen*

## GameScreen

Tela principal onde é carregado o nível e onde podemos controlar o personagem principal



Figura 46 - Imagem da tela do jogo

3/10/22, 8:47 PM GameScreen.h

```
1 #include "BaseScreen.h"
2 #include "Sprite.h"
3
4 class GameScreen : public BaseScreen
5 {
6 public:
7     void events(SDL_Event events);
8     void render();
9     void onChangeScene();
10 private:
11     std::vector<Sprite> sprites;
12 };
13
```

Figura 47 - Definições da classe GameScreen

[Criação do jogo] – Screens

```
3/10/22, 8:47 PM GameScreen.cpp
1 #include "GameScreen.h"
2 #include "World.h"
3 #include "LevelManager.h"
4 #include "Input.h"
5 #include "Game.h"
6 #include "Drawings.h"
7 #include "KeyGui.h"
8 #include "ScreenManager.h"
9 #include "Hud.h"
10 #include "Particle.h"
11
12 void GameScreen::events(SDL_Event events)
13 {
14     keyGui.events(events);
15
16     if(events.type == SDL_WINDOWEVENT)
17     {
18         if( events.window.event == SDL_WINDOWEVENT_FOCUS_LOST || events.window.event
19 == SDL_WINDOWEVENT_MOVED)
20         {
21             game.isPaused = true;
22             keyGui.setIndexZero();
23         }
24     }
25 void GameScreen::render()
26 {
27     world.update();
28     world.render();
29     hud.draw();
30
31     if(isKeyPressed(SDL_SCANCODE_ESCAPE) && !game.isPaused ){game.isPaused =
32     !game.isPaused;}
33
34     if(game.isPaused)
35     {
36         SDL_Rect pauseRect;
37         pauseRect.w = 150;
38         pauseRect.h = 90;
39         pauseRect.x = game.width/2 - pauseRect.w/2;
40         pauseRect.y = game.height/2 - pauseRect.h/2;
41         drawFilledRect(renderer,pauseRect.x, pauseRect.y, pauseRect.w, pauseRect.h,
42 {0,0,0,255});
43         keyGui.begin(pauseRect.x+47,pauseRect.y+30);
44         if(keyGui.button("CONTINUE")){game.isPaused = false;}
45         if(keyGui.button("EXIT")) {
46             game.isPaused = false;
47             screenManager.changeScreen(EScreen::TITLE_SCREEN);
48
49         }
50     }
51
52     if(world.numPlayers <= 0){
53         game.isPaused = false;
54         hud.lifes -= 1;
55         hud.lifes <= 0 ? screenManager.changeScreen(EScreen::SUBMIT_SCORE_SCREEN) :
56         screenManager.changeScreen(EScreen::GAME_OVER_SCREEN);
57 }
```

*Figura 48 - Código da classe GameScreen*

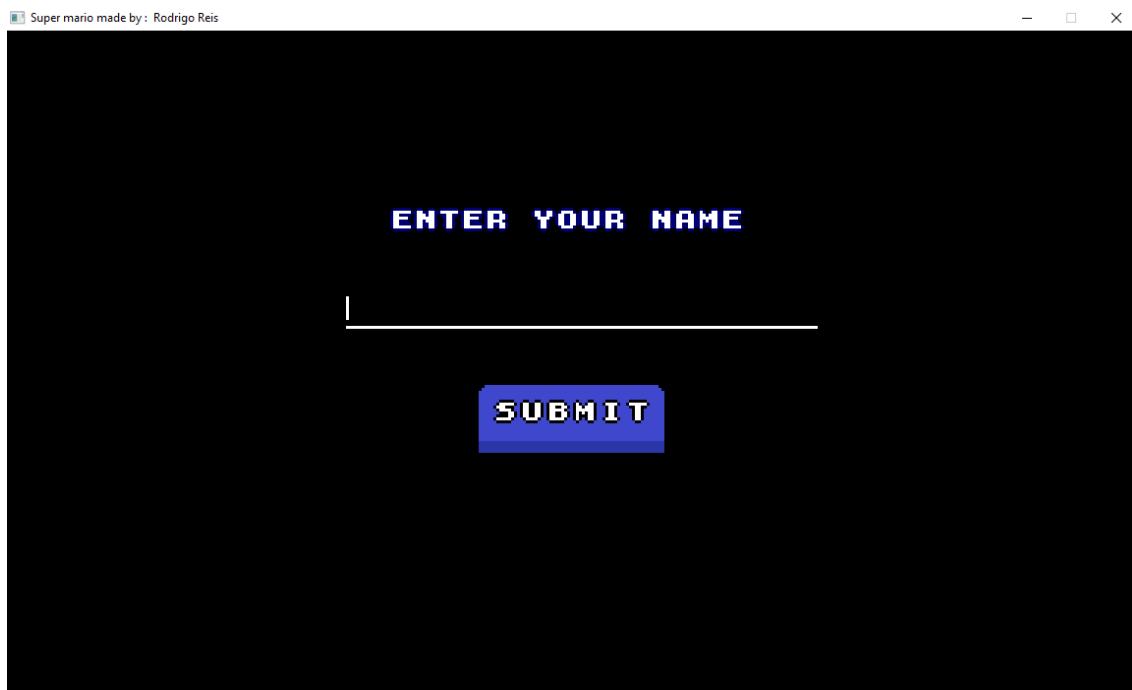
```
3/10/22, 8:47 PM GameScreen.cpp
56     }
57 }
58 void GameScreen::onChangeScene()
59 {
60     keyGui.setIndexZero();
61     keyGui.icon = &screenManager.sprites[1];
62     char levelName[100];
63     sprintf(levelName, "Resources/Levels/Level%d.txt", levelManager.currentLevel + 1);
64     levelManager.loadLevel(levelName);
65 }
66 }
```

*Figura 49 - Código da classe GameScreen*

### SubmitScoreScreen

Esta tela aparece quando as vidas chegam a zero ou quando se completa o ultimo nível e serve para guardar pontuação na tabela de pontuações.

Neste menu tem uma caixa de texto onde escrevemos o nome e um botão para submeter o formulario e guardar na tabela de pontuação.



```
3/10/22, 10:44 PM                                SubmitScoreScreen.h
1 #include "BaseScreen.h"
2 #include "Sprite.h"
3
4 class SubmitScoreScreen : public BaseScreen
5 {
6 public:
7     void events(SDL_Event events);
8     void render();
9     void onChangeScene();
10 private:
11 };
12
```

*Figura 50 - Definições da classe SubmitScore*

## [Criação do jogo] – Screens

```
3/10/22, 10:45 PM                               SubmitScoreScreen.cpp
1 #include "SubmitScoreScreen.h"
2 #include "ScreenManager.h"
3 #include "CustomFont.h"
4 #include "Button.h"
5 #include "ScoreSave.h"
6 #include "TextBox.h"
7 #include "SaveSystem.h"
8 #include "Hud.h"
9 #include "Logger.h"
10
11 Button* submitButton = nullptr;
12 TextBox submitTextbox;
13
14 void SubmitScoreScreen::events(SDL_Event events){submitTextbox.events(events);}
15
16 void SubmitScoreScreen::render()
17 {
18     defaultFont.draw(130, 60, "ENTER YOUR NAME :");
19     submitTextbox.draw();
20     if(submitButton->draw()){
21         if(submitTextbox.text.length() > 0){
22             scoreSave.saveOnline((char*)submitTextbox.text.c_str(),hud.getScore());
23             scoreSave.saveLocal((char*)submitTextbox.text.c_str(),hud.getScore());
24             screenManager.changeScreen(TITLE_SCREEN);
25         }
26     }
27 }
28
29 void SubmitScoreScreen::onChangeScene()
30 {
31     saveSystem.del(saveSystem.currentSaveSlot);
32     if(submitButton ==nullptr)
33         submitButton = new Button(Vec2(160,120), "Resources/GUI/Bt_submit.png",
34 "Resources/GUI/Bt_submit_hoverED.png");
35     submitTextbox = TextBox(115, 90, &defaultFont, 20);
36     log("game over\n");
37 }
```

*Figura 51 - Código da classe SubmitScore*

## GUI

### CustomFont

Como não gostei do facto do SDL\_ttf carregar algumas fontes sem anti-aliasing eu criei meu próprio carregador de fontes.

Ele carrega a fontes a partir de atlas com todas as letras e de um dicionário com as letras.



*Figura 52 - Fonte normal do jogo*

```
11/03/22, 09:58                                         CustomFont.h
1 #pragma once
2 #include <vector>
3 #include "Sprite.h"
4 #include <string>
5 class CustomFont
6 {
7 public:
8     std::vector<Sprite> LetterImg = std::vector<Sprite>();
9     std::string AllLetters = "";
10    int LetterWitdh = 0, LetterHeight = 0;
11    CustomFont(){}
12    CustomFont(std::string Letters, std::string pathImage, int width, int height);
13    int getTextW(std::string name) { return LetterWitdh * name.length(); }
14    int getTextH() { return LetterHeight; }
15    void draw(int x, int y, std::string text);
16 };
17
18 extern CustomFont defaultFont;
```

*Figura 53 - Definições da classe CustomFont*

```
11/03/22, 09:59                                         CustomFont.cpp
1 #include "Customfont.h"
2 #include <SDL2/SDL_image.h>
3 #include "Game.h"
4
5 CustomFont defaultFont;
6
7 CustomFont::CustomFont(std::string Letters, std::string pathImage, int width, int
height)
8 {
9     AllLetters = Letters;
10    LetterWitdh = width;
11    LetterHeight = height;
12    loadSpriteSheet(LetterImg, pathImage.c_str());
13 }
14
15 void CustomFont::draw(int x, int y, std::string text)
16 {
17     for (int i = 0; i < (int)text.length(); i++)
18     {
19         auto index = AllLetters.find(text[i]);
20
21         if (index == std::string::npos)
22             continue;
23
24         LetterImg[index].draw(x + i * LetterWitdh, y);
25     }
26 }
```

*Figura 54 - Código da classe CustomFont*

## Botão

Quando tenho mouse por cima dele o botão fica mais escuro e muda o cursor do mouse.

Se clicar nele enquanto tenho mouse em cima do botão a função **draw** retorna true.



*Figura 55 - Botão normal*



*Figura 56 - Botão quando o mouse está por cima*

```
11/03/22, 09:51                                         Button.h
1 #pragma once
2 #include "Sprite.h"
3 #include "Vec2.h"
4 class Button
5 {
6     public:
7         Vec2 position;
8         Button(){}
9         Button(Vec2 pos,const char* buttonSpritePath,const char*
buttonSpriteHoverPath);
10    bool draw();
11    private:
12    bool isMouseOver();
13    Sprite * buttonSprite;
14    Sprite * buttonSpriteHover;
15 };
16 }
```

*Figura 57 - Definições da classe botão*

```
11/03/22, 09:51                                         Button.cpp
1 #include "Input.h"
2 #include "Button.h"
3 #include <SDL2/SDL.h>
4
5 Button::Button(Vec2 pos,const char* buttonSpritePath,const char*
buttonSpriteHoverPath)
6 {
7     this->position = pos;
8     this->buttonSprite = new Sprite(buttonSpritePath);
9     this->buttonSpriteHover = new Sprite(buttonSpriteHoverPath);
10 }
11
12 bool Button::isMouseOver()
13 {
14     Vec2 mousePos = getMousePos();
15     Vec2 size = Vec2(buttonSprite->w,buttonSprite->h);
16     return (mousePos.x > position.x && mousePos.x < position.x + size.x && mousePos.y
> position.y && mousePos.y < position.y + size.y);
17 }
18
19 bool Button::draw()
20 {
21     if(isMouseOver()){
22         buttonSpriteHover->draw(position.x,position.y);
23         SDL_SetCursor(SDL_CreateSystemCursor(SDL_SYSTEM_CURSOR_HAND));
24         if(isLeftMouseClicked()){
25             return true;
26         }
27         return false;
28     }
29     buttonSprite->draw(position.x,position.y);
30     SDL_SetCursor(SDL_CreateSystemCursor(SDL_SYSTEM_CURSOR_ARROW));
31     return false;
32 }
```

*Figura 58 - Código da classe botão*

## Textbox

Uma caixa de texto que permite escrever, apagar letras e mover o cursor da caixa de texto.



Figura 59 - Caixa de texto

14/03/22, 09:02  
TextBox.h

```
1 #pragma once
2 #include "CustomFont.h"
3 #include <SDL2/SDL.h>
4
5 class TextBox {
6 public:
7     int x, y;
8     CustomFont* font;
9     std::string text;
10    int maxLenght;
11    unsigned int cursorPos = 0;
12    bool isUsing = false;
13    int lastTimeUsed = 0;
14    #define IGNORE LETTERS SIZE 56
15    int ignoreLetters[IGNORE LETTERS SIZE] =
16    {
17        SDLK_LSHIFT,
18        SDLK_RSHIFT,
19        SDLK_CAPSLOCK,
20        SDLK_DOWN,
21        SDLK_UP,
22        SDLK_LCTRL,
23        SDLK_KP_ENTER
24    };
25    TextBox() {}
26    TextBox(int x, int y, CustomFont* font, int maxLength);
27    void events(SDL_Event event);
28    void draw();
29 private:
30 };
31
32 };
33 }
```

Figura 60 - Definições da classe TextBox

## [Criação do jogo] – GUI

```
14/03/22, 09:03          TextBox.cpp

1 #include "Textbox.h"
2 #include "Drawings.h"
3 #include "Game.h"
4
5 void TextBox::events(SDL_Event event)
6 {
7     if (event.type == SDL_KEYDOWN)
8     {
9         isUsing = true;
10        lastTimeUsed = SDL_GetTicks();
11        int index = event.key.keysym.sym;
12        switch (index)
13        {
14            case SDLK_RIGHT:
15                if(cursorPos < text.length())
16                    cursorPos++;
17                break;
18            case SDLK_LEFT:
19                if(cursorPos > 0)
20                    cursorPos--;
21                break;
22            case SDLK_BACKSPACE:
23                if (text.length() > 0 && cursorPos > 0)
24                {
25                    text.erase(cursorPos-1, 1);
26                    cursorPos--;
27                }
28                break;
29            default:
30                if (text.length() < maxLenght)
31                {
32                    for (int i = 0; i < IGNORE LETTERS_SIZE; i++)
33                    {
34                        if (index == ignoreLetters[i]) {
35                            return;
36                        }
37                    }
38                    char letter = (char)toupper(index);
39                    text.insert(cursorPos,1,letter);
40                    cursorPos++;
41                }
42                break;
43            }
44        }
45    }
46 }
47
48
49 void TextBox::draw()
50 {
51     //draw bottom-border
52     drawFilledRect(renderer, x, y + font->getTextH() + 2, maxLenght * font-
>LetterWitchd,1,{255, 255,255,255});
53     //draw text
54     font->draw(x, y, text);
55
56     if (SDL_GetTicks() > lastTimeUsed + 500)
57     {
58         int drawCursor = (SDL_GetTicks() / 600) % 2;
```

## [Criação do jogo] – GUI

```
14/03/22, 09:03                                         TextBox.cpp

59
60     if (drawCursor == 1)
61         drawFilledRect(renderer, x + font->LetterWitch * cursorPos, y, 1, font-
|>getTextH(),{255, 255, 255});
62     }
63     else {
64         drawFilledRect(renderer, x + font->LetterWitch * cursorPos, y, 1, font-
|>getTextH(), {255, 255, 255});
65     }
66 }
67
68 TextBox::TextBox(int x, int y, CustomFont* _font, int maxlenht)
69 {
70     this->x = x;
71     this->y = y;
72     this->font = _font;
73     this->maxLenght = maxlenht;
74     this->cursorPos = 0;
75 }
```

*Figura 61 - Código da classe TextBox*

## Hud

O hud mostra a pontuação atual, numero de moedas, nome do nível e tempo que resta para completar o nível.



*Figura 62 - Imagem do Hud*

14/03/22, 09:17 Hud.h

```

1 #include <string>
2 #include <vector>
3 #include "Sprite.h"
4 class Hud
5 {
6 public:
7     Hud();
8     ~Hud();
9
10    void init();
11    void draw();
12    void addCoin(int amount);
13    void addScore(int amount);
14    void setCoin(int amount);
15    void setScore(int amount);
16
17    int getScore() { return score; }
18    int getCoin() { return coin; }
19
20    int lives = 5;
21    float countdownTimer = 500;
22    float countdownTimerMax = 500;
23 private:
24     std::string coinStr = "x00";
25     std::string scoreStr = "000000";
26     std::vector<Sprite> coinSprites;
27
28     int coin = 0;
29     int score = 0;
30 };
31
32 extern Hud hud;

```

*Figura 63 - Definições da classe Hud*

## [Criação do jogo] – GUI

```
14/03/22, 09:18                                     Hud.cpp

1 #include "Hud.h"
2 #include "Utils.h"
3 #include "CustomFont.h"
4 #include "Timer.h"
5 #include "LevelManager.h"
6 #include "ScreenManager.h"
7 #include "GameOverScreen.h"
8 #include "Game.h"
9 #include "Logger.h"
10
11 #define LIMIT_COIN 99
12 #define LIMIT_SCORE 999999
13
14 Hud hud = Hud();
15
16 void Hud::init()
17 {
18     loadSpriteSheet(coinSprites, "Resources/Fonts/HudCoin.png");
19 }
20
21 void Hud::draw()
22 {
23     if(countdownTimer > countdownTimerMax){
24         countdownTimer = countdownTimerMax;
25     }
26
27     int x = 80;
28     int y = 17;
29
30     defaultFont.draw(x,y, "MARIO");
31     defaultFont.draw(x,y+11, scoreStr);
32
33     int coinIndex = ANIMATION(0,3,300);
34     coinSprites[coinIndex].draw(x+65,y+11);
35     defaultFont.draw(x+65+10,y+11, coinStr);
36
37     defaultFont.draw(x+115,y, "WORLD");
38     int levelW = defaultFont.getTextW(levelManager.displayName);
39     defaultFont.draw(x+113 + levelW/2,y+11,levelManager.displayName);
40
41
42     defaultFont.draw(x+184,y, "TIME");
43     defaultFont.draw(x+184+8,y+11, std::to_string((int)countdownTimer));
44
45     if(!game.isPaused){
46         countdownTimer -= timer.deltaTime;
47     }
48
49     if(countdownTimer < 0){
50         countdownTimer = 0;
51         gameOverType = EGameOverType::TimeUp;
52         hud.lifes -= 1;
53         hud.lifes <= 0 ? screenManager.changeScreen(EScreen::SUBMIT_SCORE_SCREEN) :
54             screenManager.changeScreen(EScreen::GAME_OVER_SCREEN);
55     }
56 }
57
58 void Hud::addCoin(int amount)
```

*Figura 64 - Código da classe Hud*

## [Criação do jogo] – GUI

```
14/03/22, 09:18                                     Hud.cpp
59 {
60     if (this->coin == LIMIT_COIN)
61         return;
62
63     this->coin += amount;
64
65     coinStr = "x";      //Coin to string format -> x01
66     if (this->coin < 10)
67         coinStr += "0";
68     coinStr += std::to_string(this->coin);
69 }
70
71 void Hud::setCoin(int amount)
72 {
73     if (this->coin == LIMIT_COIN)
74         return;
75
76     this->coin = amount;
77     coinStr = "x";
78     if (this->coin < 10)
79         coinStr += "0";
80     coinStr += std::to_string(this->coin);
81 }
82
83 void Hud::addScore(int amount)
84 {
85     if (this->score == LIMIT_SCORE)
86         return;
87
88     this->score += amount;
89     log("Score: %d\n", this->score);
90     int size = intLength(this->score);      //Score to string format 000100
91     scoreStr = "";
92     for (int i = 0; i < 6 - size; i++){
93         scoreStr += "0";
94     }
95     scoreStr += std::to_string(this->score);
96 }
97
98 void Hud::setScore(int amount){
99     this->score = amount;
100    scoreStr = "";
101    int size = intLength(this->score);
102    for (int i = 0; i < 6 - size; i++)
103    {
104        scoreStr += "0";
105    }
106    scoreStr += std::to_string(this->score);
107 }
```

*Figura 65 - Código da classe Hud*

**Table**

Esta classe permite criar uma tabela, adicionar n colunas, items as colunas(linhas) e permite também desenhar a bordas se eu quiser.

1.	JOAO	2424242
2.	JOAO	024242
3.	MANUEL	021012
4.	VASCO	212021
5.	DIOGO	212021
6.	PEDRO	126123
7.	INACIO ?	55551
8.	NOME	00200

Figura 66 - Imagem de uma tabela criada

```
15/03/22, 08:56                                         Table.h
1 #pragma once
2 #include "Vec2.h"
3 #include <string>
4 #include <vector>
5 #include <SDL2/SDL.h>
6 class Collum
7 {
8     public:
9     Collum(std::string name, int w)
10    {
11         this->name = name;
12         this->w = w;
13     }
14     std::string name = "";
15     int w = 0;
16 };
17
18 class Table
19 {
20 public:
21     Table() {}
22     SDL_Color backgrondColor = { 0,0,0,255 };
23     int x = 0, y = 0;
24     std::vector<Collum> cols = std::vector<Collum>();
25     std::vector<std::string> items = std::vector<std::string>();
26
27     bool drawBorder = false;
28     bool drawHeader = false;
29     int headerSize = 20;
30     int colSize = 20;
31
32     void addCollum(std::string name, int w);
33     void addItem(std::string value);
34     void draw();
35
36 private:
37 };
```

Figura 67 - Definições da classe Table

## [Criação do jogo] – GUI

```
15/03/22, 08:57                                         Table.cpp
1 #include "Table.h"
2 #include "Game.h"
3 #include "Drawings.h"
4 #include "CustomFont.h"
5
6 void Table::addCollum(std::string name, int w) {this->cols.push_back({ name,w });}
7
8 void Table::addItem(std::string value) {this->items.push_back(value);}
9
10 void Table::draw()
11 {
12     //int thickness = 1;
13     //if (!drawBorder) { thickness = 0; }
14     if (!drawHeader) { headerSize = 0; }
15     //DrawFilledRect(renderer, x, y, 0, 0, backgrondColor.r, backgrondColor.g,
16     backgrondColor.b, backgrondColor.a);
17     int TotalX = 0;
18     for (auto c : this->cols)
19     {
20         //drawRect(x + TotalX, y, c.w, colSize, thickness,{255, 0, 0});
21         int textLength = defaultFont.getTextW(c.name);
22
23         defaultFont.draw(
24             x + TotalX + c.w / 2 - (textLength / 2), //centeredX
25             y + colSize / 2 - defaultFont.getTextH() / 2, //centeredY
26             c.name);
27
28         TotalX += c.w;
29     }
30     int MarginLeftItem = 5;
31     //Draw items
32     TotalX = 0;
33     int TotalY = 0;
34     int TotalCols = 0;
35     for (int i = 0; i < (int)items.size(); i++)
36     {
37         //int ItemX = x + TotalX;
38         //int ItemY = y + TotalY + colSize;
39         //drawRect(renderer, ItemX, ItemY, cols[TotalCols].w, colSize, thickness, 255, 0,
40         0, 255); //border
41         defaultFont.draw(
42             x + TotalX + MarginLeftItem,
43             y + TotalY + headerSize + colSize / 2 - defaultFont.getTextH() / 2
44             , items[i]
45         );
46
47         TotalX += cols[TotalCols].w;
48         TotalCols++;
49         if (TotalCols == (int)cols.size())
50         {
51             TotalY += colSize;
52             TotalX = 0;
53             TotalCols = 0;
54         }
55 }
```

Figura 68 - Código da classe Table

## Entities

### Ent

Classes que é herdada por todas as entidades que não se movem.

**onUpdate** - Função chamada todos os frames do jogo onde normalmente tem movimentação

**onColide** - Chamada quando outro objeto que move colide.(virtual significa que pode ser alterado pela classe que herda da Ent.cpp).

**ondraw** – chama função W2S(WorldToScreen) da câmera e desenha no ecrã se coordenada estiver dentro do limite da janela.

**onDead** – Chamado quando a entidade morre,na maior parte das entidades custumo mudar sprite atual para sprite de morto e adiciono X de score e dou spawn PopUp(que é número de pontos que personagem ganhou e sobe a uma velocidade constante e depois de certo tempo desaparece ).

## [Criação do jogo] – Entities

```
15/03/22, 09:11 Ent.h

1 #pragma once
2 #include "Vec2.h"
3 #include "Sprite.h"
4 #include <vector>
5 enum ETags{
6     TEnt = 0,
7     TController,
8     TShell,
9     TBowser,
10    TFireball,
11    TToad,
12    TSack
13 };
14
15 enum ECol{
16     Trigger,
17     Solid
18 };
19
20 enum ELayers
21 {
22     LNONE = 0,
23     LFIREBALCHAIN = 1,
24     LPLAYER,
25     LTUBE,
26     LENEMY,
27     LPowerUp,
28 };
29
30 struct Colider{
31     ECol type;
32     Vec2 offset; //relative to the ent
33     Vec2 size;
34 };
35
36 class Ent
37 {
38     public:
39
40     //Entity position
41     Vec2 pos = Vec2(0,0);
42
43     //Identifies the object's class
44     unsigned short tag = ETAGS::TEnt;
45
46     //Layer determines whether to draw over an object or not
47     unsigned short layer = 0;
48
49     //Identify if the Ent is dead or alive
50     bool isAlive = true;
51
52     SDL_RendererFlip flip = SDL_FLIP_NONE;
53
54     //Colliders for the Ent
55     std::vector<Colider> coliders = std::vector<Colider>();
56
57     Sprite* sprite;
58
59     virtual ~Ent(){}
}
```

Figura 69 - Definições da classe Ent

## [Criação do jogo] – Entities

```
15/03/22, 09:11 Ent.h
60     Ent() {};
61     Ent(float x, float y) : pos(x, y) {};
62     Ent(Vec2 pos, int tag, int layer, Sprite* sprite);
63     Ent(int layer, Sprite* sprite, ECol ecol);
64     Ent(int layer, Sprite* sprite){this->layer = layer; this->sprite = sprite;}
65
66     void drawColiders(Vec2 out);
67     virtual void onStart() {};
68     virtual void onUpdate() {};
69     virtual void onRender();
70     virtual void onDead() {};
71     virtual bool onColide(Ent* ent, int colIndex) { return true; } //return true if u
| want resolution
72         virtual Ent* clone(); //PROTOTYPE DESIGN PATTERN
73 
```

**Figura 70 - Definições da classe Ent**

```
15/03/22, 09:20 Ent.cpp
1 #include "Ent.h"
2 #include "Drawings.h"
3 #include "Game.h"
4 #include "World.h"
5 //Draw coliders
6 //#define DRAW_DEBUG
7 Ent::Ent(Vec2 pos, int tag, int layer, Sprite* sprite)
8 {
9     this->pos = pos;
10    this->tag = tag;
11    this->layer = layer;
12    this->sprite = sprite;
13 }
14
15 void Ent::onRender(){
16
17     Vec2 out = {0,0};
18     if(world.currentCamera != NULL){
19         if(world.currentCamera->W2S(this, out)){
20             this->drawColiders(out);
21             this->sprite->draw(out.x,out.y,flip);
22         }
23     }
24     else{
25         out = this->pos;    this->drawColiders(out);
26         this->sprite->draw(out.x,out.y,flip);
27     }
28
29 }
30
31 Ent::Ent(int layer, Sprite* sprite, ECol ecol)
32 {
33     this->layer = layer;
34     this->sprite = sprite;
35     this->coliders.push_back({ ecol, Vec2(0, 0), Vec2(sprite->w, sprite->h) });
36 }
37
38 void Ent::drawColiders(Vec2 out)
39 {
40     #ifndef DRAW_DEBUG
41         return;
42     #endif
43
44     for (int i = 0; i < (int)coliders.size(); i++)
45     {
46         drawRect(renderer,
47                 coliders[i].offset.x + out.x,
48                 coliders[i].offset.y + out.y,
49                 coliders[i].size.x,
50                 coliders[i].size.y, 1,
51                 {coliders[i].type*255, 0, 0, 255});
52     }
53 }
54 Ent* Ent::clone(){return new Ent(*this);}
55 
```

**Figura 71 - Código classe Ent**

## LivingEnt

Classe que é herdada por todas as entidades que se movem.

Esta classe tem código da Ent mais a deteção de colisão e as variáveis de velocidade.

**checkColisionY** - Detecta se está a colidir com algum objeto no eixo Y e se estiver teleporta para trás.

**checkColisionX** - Detecta se está colidir com algum objeto no eixo X e se estiver teleporta para trás e põe velocidade a 0 para não ultrapassar o chão.

Nota: Para saber onde está a colidir uso a velocidade da LivingEnt, por exemplo se a velocidade for superior a 0 quer dizer estou mover para direita e sei que estou bater no lado esquerdo bloco.

```
15/03/22, 09:25                                         LivingEnt.h
1 #pragma once
2 #include "Ent.h"
3 #include "Vec2.h"
4 class LivingEnt : public Ent
5 {
6     public:
7         LivingEnt(){};
8         LivingEnt(int layer, Sprite* sprite, ECol ecol) : Ent(layer, sprite, ecol){};
9         Vec2 velocity = Vec2(0,0);
10        bool isGrounded = false;
11        bool noColision = false;
12        bool checkCollisionX();
13        bool checkCollisionY();
14    private:
15};
```

*Figura 72 - Definições da classe LivingEnt*

## [Criação do jogo] – Entities

```
15/03/22, 12:54                                         LivingEnt.cpp
1 #include "LivingEnt.h"
2 #include "World.h"
3
4 bool LivingEnt::checkCollisionX()
5 {
6     if(noColision){return false;}
7     bool result = false;
8     for(int colIndex = 0; colIndex < this->coliders.size(); colIndex++)
9     {
10         if(coliders[colIndex].type == ECol::Trigger){continue;}
11
12         SDL_Rect myRect = {
13             (int)(pos.x + coliders[colIndex].offset.x),
14             (int)(pos.y + coliders[colIndex].offset.y),
15             (int)coliders[colIndex].size.x,
16             (int)coliders[colIndex].size.y
17         };
18
19         for(int i = 0; i < world.getEntityCount(); i++)
20         {
21             Ent* ent = world.getEnt(i);
22
23             if(ent == this){continue;}
24
25             for(int col = 0; col < (int)ent->coliders.size(); col++)
26             {
27                 SDL_Rect rect = {
28                     (int)(ent->pos.x + ent->coliders[col].offset.x),
29                     (int)(ent->pos.y + ent->coliders[col].offset.y),
30                     (int)ent->coliders[col].size.x,
31                     (int)ent->coliders[col].size.y
32                 };
33
34                 if(SDL_HasIntersection(&myRect,&rect))
35                 {
36                     bool doResolution = ent->onColide(this,col);
37
38                     if(doResolution == false){continue;}
39
40                     if(ent->coliders[col].type == ECol::Trigger){ continue;}
41
42                     if(this->velocity.x > 0){ //moving right
43                         this->pos.x = ent->pos.x - this->coliders[col].size.x;
44                     }
45                     else if(this->velocity.x < 0){ //moving left
46                         this->pos.x = ent->pos.x + ent->coliders[col].size.x;
47                     }
48                     this->velocity.x = 0;
49                     result = true;
50                 }
51             }
52         }
53     }
54     return result;
55 }
56
57 bool LivingEnt::checkCollisionY()
58 {
59     if(noColision){return false;}
```

Figura 73 - Código da LivingEnt

## [Criação do jogo] – Entities

```
15/03/22, 12:54                                         LivingEnt.cpp
60     bool result = false;
61     for(int colIndex = 0; colIndex < this->coliders.size(); colIndex++)
62     {
63         if(coliders[colIndex].type == ECol::Trigger){continue;}
64
65         SDL_Rect myRect = {
66             (int)(pos.x + coliders[colIndex].offset.x),
67             (int)(pos.y + coliders[colIndex].offset.y),
68             (int)coliders[colIndex].size.x,
69             (int)coliders[colIndex].size.y
70         };
71
72         for(int i = 0; i < world.getEntityCount(); i++)
73         {
74             Ent* ent = world.getEnt(i);
75
76             //check if ent is a controller
77             if(ent == this){continue;}
78
79             for(int col = 0; col < (int)ent->coliders.size(); col++)
80             {
81                 SDL_Rect rect = {
82                     (int)(ent->pos.x + (int)ent->coliders[col].offset.x),
83                     (int)(ent->pos.y + (int)ent->coliders[col].offset.y),
84                     (int)ent->coliders[col].size.x,
85                     (int)ent->coliders[col].size.y
86                 };
87
88                 if(SDL_HasIntersection(&myRect,&rect))
89                 {
90                     bool doResolution = ent->onColide(this,col);
91
92                     if(doResolution == false){continue;}
93                     if(ent->coliders[col].type == ECol::Trigger ){ continue;}
94
95
96                     if(this->velocity.y > 0){ //moving down
97                         this->pos.y = ent->pos.y - this->coliders[col].size.y;
98                         isGrounded = true;
99                     }
100                    else if(this->velocity.y < 0){ //moving up
101                        this->pos.y = ent->pos.y + ent->coliders[col].size.y +1;
102                        isGrounded = false;
103                    }
104                    this->velocity.y = 0;
105                    result = true;
106                }
107            }
108        }
109    }
110
111 }
```

*Figura 74 - Código da classe LivingEnt*

## MysteryBox

É um bloco que tem dois estados quanto esta com item onde faz animação Sprite 1 ate 4 e quando esta vazio que muda para ultimo Sprite.

Este bloco pode conter vario tipos itens (cogumelo, flor, moedas...).

Quando o controller colide na parte baixo da MysteryBox a mesma fica vazia, da spawn ao item que esta a guardar.



*Figura 75 - Imagem da spriteSheet da MysteryBox*

```
15/03/22, 13:18                                         MysteryBox.h
1 #pragma once
2 #include "Ent.h"
3
4 enum EHoldItem
5 {
6     EHoldNone = -1,
7     EHoldCoin,
8     EHoldMushroom,
9     EHoldFlower,
10    EHoldStar
11 };
12
13 class MysteryBox : public Ent
14 {
15 public:
16     MysteryBox(Sprite* sprite,EHoldItem holdItem) : Ent(0,sprite,EColl::Solid) {
17         this->holdItem = holdItem;
18     };
19     void onRender();
20     void onStart();
21     void onUpdate();
22     bool onColide(Ent* ent,int colIndex);
23     Ent* clone() { return new MysteryBox(*this); }
24 private:
25     bool isEmpty = false;
26     bool shakeUp = false;
27     bool shakeDown = false;
28     EHoldItem holdItem = EHoldItem::EHoldNone;
29
30     float elapsedTime = 0.0f;
31     float startY = 0;
32     Sprite* startSprite;
33 };
34
```

*Figura 76 - Definições da classe MysteryBox*

## [Criação do jogo] – Entities

```
15/03/22, 13:19                               MysteryBox.cpp
1 #include "MysteryBox.h"
2 #include "LivingEnt.h"
3 #include "Timer.h"
4 #include "Utils.h"
5 #include "Hud.h"
6 #include "Particle.h"
7 #include "Prefabs.h"
8 #include "World.h"
9 #include "SoundManager.h"
10#define SHAKE_DURATION 0.05f
11void MysteryBox::onStart()
12{
13    this->coliders.push_back({ ECol::Trigger, Vec2(4, sprite->h), Vec2(sprite->w-8,1) });
14    this->startSprite = sprite;
15    this->startY = pos.y;
16}
17
18void MysteryBox::onUpdate()
19{
20    if(!this->isEmpty){
21        this->sprite = this->startSprite + ANIMATION(0,3,300);
22    }
23}
24
25void MysteryBox::onRender()
26{
27    Vec2 out = {0,0};
28    bool canRender = true;
29    if(world.currentCamera != NULL){
30        canRender = world.currentCamera->W2S(this, out);
31    }
32    else{
33        out = this->pos;
34    }
35
36    if(!canRender){return;}
37
38    this->drawColiders(out);
39    Vec2 drawPos = {out.x,out.y};
40
41    if(shakeUp)
42    {
43        this->elapsedTime+= timer.deltaTime;
44        float percentage = elapsedTime / SHAKE_DURATION;
45        drawPos.y = lerp(startY,startY-4, percentage);
46
47        if(percentage >= 1.0f){
48            shakeUp = false;
49            shakeDown = true;
50            elapsedTime = 0.0f;
51        }
52    }
53
54    if(shakeDown){
55        this->elapsedTime+= timer.deltaTime;
56        float percentage = elapsedTime / SHAKE_DURATION;
57        drawPos.y = lerp(startY-4,startY, percentage);
58    }
```

Figura 77 - Código da classe MysteryBox

## [Criação do jogo] – Entities

```
15/03/22, 13:19                         MysteryBox.cpp
59
60     if(percentage >= 1.0f){
61         shakeDown = false;
62         shakeUp = false;
63         elapsedTime = 0.0f;
64     }
65 }
66
67     this->sprite->draw(drawPos.x,drawPos.y,flip);
68 }
69
70
71 bool MysteryBox::onColide(Entity* ent,int colIndex)
72 {
73     if(this->isEmpty){return true;}
74     if(ent->tag != ETAGS::TController){return true;}
75     if(colIndex != 1){return true;} //colider index 1 is bottom colider
76
77     switch (holdItem)
78     {
79         case EHoldCoin:
80         {
81             hud.addScore(200);
82
83             Particle* score200 = new Particle(&particleSprites[1],this->pos,
84             Vec2(0,-30), 0.5f, false);
85             Particle* explosionCoin = new Particle(&particleSprites[17], Vec2(0,0),
86             Vec2(0,-300), 0.1f, true);
87             Particle* coinParticle = new Particle(&particleSprites[13], Vec2(pos.x-
88             2,pos.y-6), Vec2(0,-100), 0.5f, false);
89
90             explosionCoin->childParticle = score200;
91             explosionCoin->setAnimation(4,50);
92             coinParticle->setAnimation(4,100);
93             coinParticle->childParticle = explosionCoin;
94             particleSystem.addParticle(coinParticle);
95             hud.addCoin(1);
96
97         break;
98     }
99     case EHoldMushroom:
100    {
101        world.addEntity(getPrefab(28,pos.x,pos.y - 25),world.getScene());
102        soundManager.playSFX("ItemSprout.wav");
103        break;
104    }
105    case EHoldFlower:
106    {
107        world.addEntity(getPrefab(29,pos.x,pos.y - 16),world.getScene());
108        break;
109    }
110    case EHoldStar:
111    {
112        world.addEntity(getPrefab(16,pos.x,pos.y - 16),world.getScene());
113        break;
114    }
115
116     isEmpty = true;
117 }
```

**Figura 78 - Código da classe MysteryBox**

```
15/03/22, 13:19                         MysteryBox.cpp
116     shakeUp = true;
117     this->sprite = this->startSprite+4;
118     return true;
119 }
```

**Figura 79 - Código da classe MysteryBox**

## Goomba

Goomba é dos primeiros inimigos que se encontra ao jogar com um formato semelhante a um cogumelo.

Ele anda para esquerda e quando colide inverte a direção inverte a direção.



Figura 80 - SpriteSheet do goomba



Figura 81 - Caixas de colisão do goomba



Figura 82 - Goomba no jogo

5/31/22, 12:55 PM  
Goomba.h

```

1 #pragma once
2 #include "LivingEnt.h"
3
4 class Goomba : public LivingEnt {
5
6 public:
7     ~Goomba(){}
8     Goomba(Sprite* sprite) : LivingEnt(0, sprite, ECol::Solid) {};
9     //void onRender();
10    void onStart();
11    void onUpdate();
12    bool onColide (Ent* ent,int colIndex);
13    void onDead();
14    Ent* clpne() { return new Goomba(*this); }
15
16 private:
17     Sprite* startSprite;
18     bool isFreeze = false;
19     int deadType = 0;
20     bool deadAnimation = false;
21     bool noAnimation = false;
22     int deadTime = 0;
23     int currentDir = 1;
24     float speed = 60.0f;
25     int delayToDie = 0;
26 };
27

```

Figura 83 - Definições do goomba

### Controller

Esta classe é usada para instanciar os players.

É usada para instanciar o mario e o luigi , nesta classe contem o controles do player ,movimentação , as diferente formas(small mario,medium,fire) e herda da classe living ent onde tem as funções de colisão



**Figura 84 - Diferentes formas do mario**



**Figura 85 - Diferente formas do luigi**

## [Criação do jogo] – Entities

```
5/31/22, 1:17 PM                                         Controller.h
1 #include "LivingEnt.h"
2 #include "Camera.h"
3 #include <vector>
4
5 enum ControllerSizes{
6     CSMALL = 0,
7     CMEDIUM,
8     CFIRE,
9     CSTAR
10 };
11
12 struct PlayerControls
13 {
14     int down = 0;
15     int left = 0;
16     int right = 0;
17     int jump = 0;
18     int attack = 0;
19     int run = 0;
20 };
21
22 class Controller : public LivingEnt
23 {
24     public:
25         //Pixels per second
26         float currentSpeed = 0;
27         float walkSpeed = 120.0f;
28         float runSpeed = 180.0f;
29
30         //0 -> left 1 -> right
31         int currentDir = 0;
32
33         //FLAGS
34         bool isJumping = false;
35         bool isFreeze = false;
36         bool deadAnimation = false;
37         bool godMode = false;
38         bool gravity = true;
39         bool flickering = false;
40         bool deathMusicPlayed = false;
41
42         //Jump
43         int jumpStartTime = 0;
44         int jumpLimit = 250;
45         #define jumpForce -350.0f
46
47         float elapsedTime = 0;
48         int startDeadTime = 0;
49
50         float timeShoot = 0;
51         float animationShootTime = 0.0f;
52
53         PlayerControls* controls;
54         Camera * followCamera = nullptr;
55
56         void setImmortal(float secondsImmortal);
57         void changeSize(ControllerSizes size);
58         ControllerSizes getSize();
59 }
```

Figura 86 - Definições do Controller

## [Criação do jogo] – Entities

```
5/31/22, 1:17 PM                                         Controller.h
60     void freeze(){isFreeze = true; this->velocity = Vec2(0,0);}
61     ~Controller();
62     Controller();
63     Controller(PlayerControls* controls,char* characterName);
64     bool onColide(Ent* ent,int colIndex);
65     void changeSprite(int spriteIndex);
66     void onStart();
67     void onRender();
68     void onUpdate();
69     void onDead();
70     void centerCamera();
71     void moveLeft();
72     void moveRight();
73     void takeDamage();
74 private:
75     float godModeSeconds = 0;
76     void movement();
77     void jumping();
78     void init(char* characterName);
79     std::vector<std::vector<Sprite>> sprites;
80     ControllerSizes controllerSize = CSMALL;
81 };
82
83
84 extern PlayerControls player1Controls;
85 extern PlayerControls player2Controls;
```

**Figura 87 - Definições do Controller**

[Objetivos do projeto,Interesse,Disciplinas envolvidas no projeto,Saberes,Fases temporais]

## Objetivos do projeto

- Aprendizagem de uma nova biblioteca de desenho (SDL).
- Aprendizagem da criação de GUI
- Criação de autonomia na área programação (Neste caso aprender algo do zero , criar também do zero)
- Aprender como funciona uma engine.
- Organização e criação de uma engine
- Aprender outras línguas que não são dadas na escola (infelizmente não ensinam c++ 😞)

## Interesse e aplicabilidade do projeto

Normalmente costumo dar reversing a jogos com IDA(Interactive Disassembler) e Reclass e por isso quis criar um jogo do zero em c++ para aprender melhor como funciona por detrás das cortinas, então comecei fazer recriação do mario , aproveitei e usei como projeto para pap.

Em termos de aplicabilidade podem analisar o código que tenho e criar uma engine melhor talvez com editor e também pode ser usado como fonte de entretenimento.

## Disciplina envolvidas no projeto

Neste projeto foi a utilizado a disciplina de Programação onde aprendemos uma linguagem parecida ao c++ chamada de c#.

## Saberes e competências profissionais

Na criação do jogo maioritariamente dos conhecimentos eu aprendi em casa e em alguns vídeos de teoria, também aprendi a linguagem C# na escola onde adquiri conhecimentos de novos algoritmos como BubbleSort,A\* entre outros.

## Fases temporais do projeto

- Protótipo - 2020/2020
- Planeamento - 2021/2021
- Aquisição do material - 2020/2021
- Preparação do ambiente trabalho - 2021/2021
- Criação do jogo - 2021/2022

## Horas previstas na implementação do projeto

As horas previstas na criação do jogo são apenas estimativas e não as horas exatas .

- Protótipo - 100 Horas
- Planeamento – 10 Horas
- Aquisição do material - 50 Horas
- Preparação do ambiente trabalho - 30 minutos
- Criação do jogo - 500 Horas

## Recursos Humanos envolvidos

Os recursos humanos neste projeto são:

- Um aluno
- Videos de pessoas da internet

## Recursos Materiais envolvidos.

Os recursos materiais neste projeto são:

- Meu Computador torre – 1000 euros
- Computadores da escola
- Word – 40 euros
- Acesso a internet – 60 a mês (12 meses) = 720 euros
- Windows 10 – versão gratuita
- Visual studio 2019 - gratuito
- Visual studio code (para protótipo) – gratuito
- Photopea(photoshop online) - gratuito
- Paint - gratuito
- Online png tools - gratuito (<https://onlinepngtools.com/change-png-color>)

## Custos Estimados do projeto

O custo total do projeto é equivalente a 1760 euros , a escolha dos materiais foi feita a maneira de ser o mais eficiente e barato possível.

## Reflexão sobre a exequibilidade do projeto.

Visto que matérias estão todos disponíveis e que tenho o que preciso para fazer o jogo, projeto será realizado dentro dos prazos estipulados

## Parceria

Este projeto não é realizado com nenhuma parceria

## Bibliografia

<https://www.spriters-resource.com/> – Foi buscar as imagens(sprites) (18/10/2021)

<https://downloads.khinsider.com/game-soundtracks/album/super-mario-all-stars-super-mario-bros> - Foi buscar as musicas do Mário(20/10/2021)

<https://www.superluigibros.com/super-mario-allstars-sounds-wav> - Foi buscar os sounds Effects(20/10/2021)

[https://www.youtube.com/watch?v=PrDdfALWH\\_4&t=0s](https://www.youtube.com/watch?v=PrDdfALWH_4&t=0s) - Como Incluir o SDL(18/10/2021)

<https://www.youtube.com/watch?v=0TIVpiQbFiE&t=85s> - Como Inclui o SDLImage(20/10/2021)

[https://www.youtube.com/watch?v=a\\_YTkIVVNoQ](https://www.youtube.com/watch?v=a_YTkIVVNoQ) - Aprendi sobre colisão em jogos 2D(18/10/2021)

<https://www.libsdl.org/download-2.0.php> - Onde foi buscar o SDL(18/10/2021)

[https://www.libsdl.org/projects/SDL\\_mixer/](https://www.libsdl.org/projects/SDL_mixer/) - Onde foi buscar o SDLMixer(20/10/2021)