Fundamentos de Programação 2

Projeto: Super Mario Bros

Autores: Walter Hugo e Gabriel Schultz

Prof.: Jean M.Simão

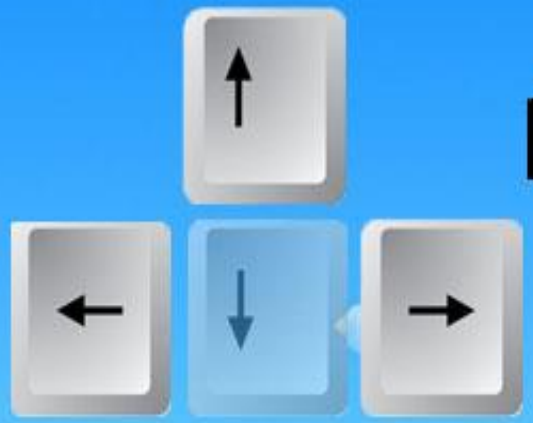
Data:02/12/2015


SUPER MARIO
BROS.
1 JOGAR
2 INSTRUCOES
RECORDS
ESC SAIR

(1) PLAYER

(2) PLAYERS

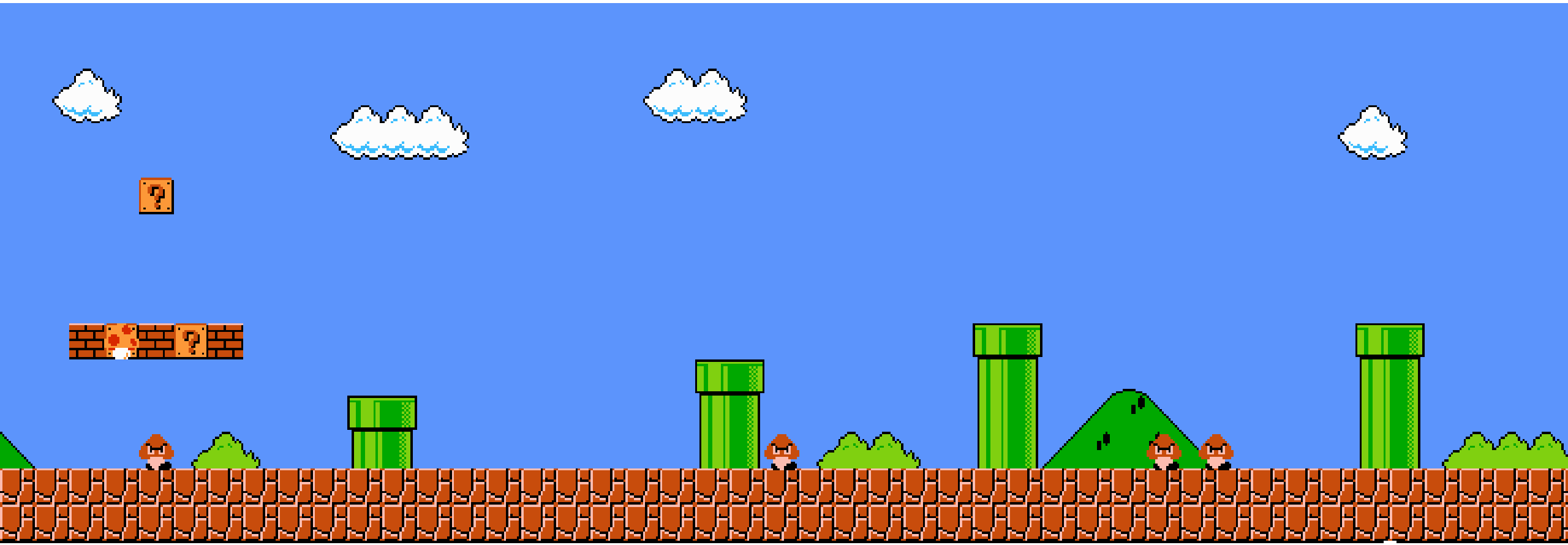
PULO



DIREITA

VOLTAR
/PAUSAR JOGO

ESQUERDA

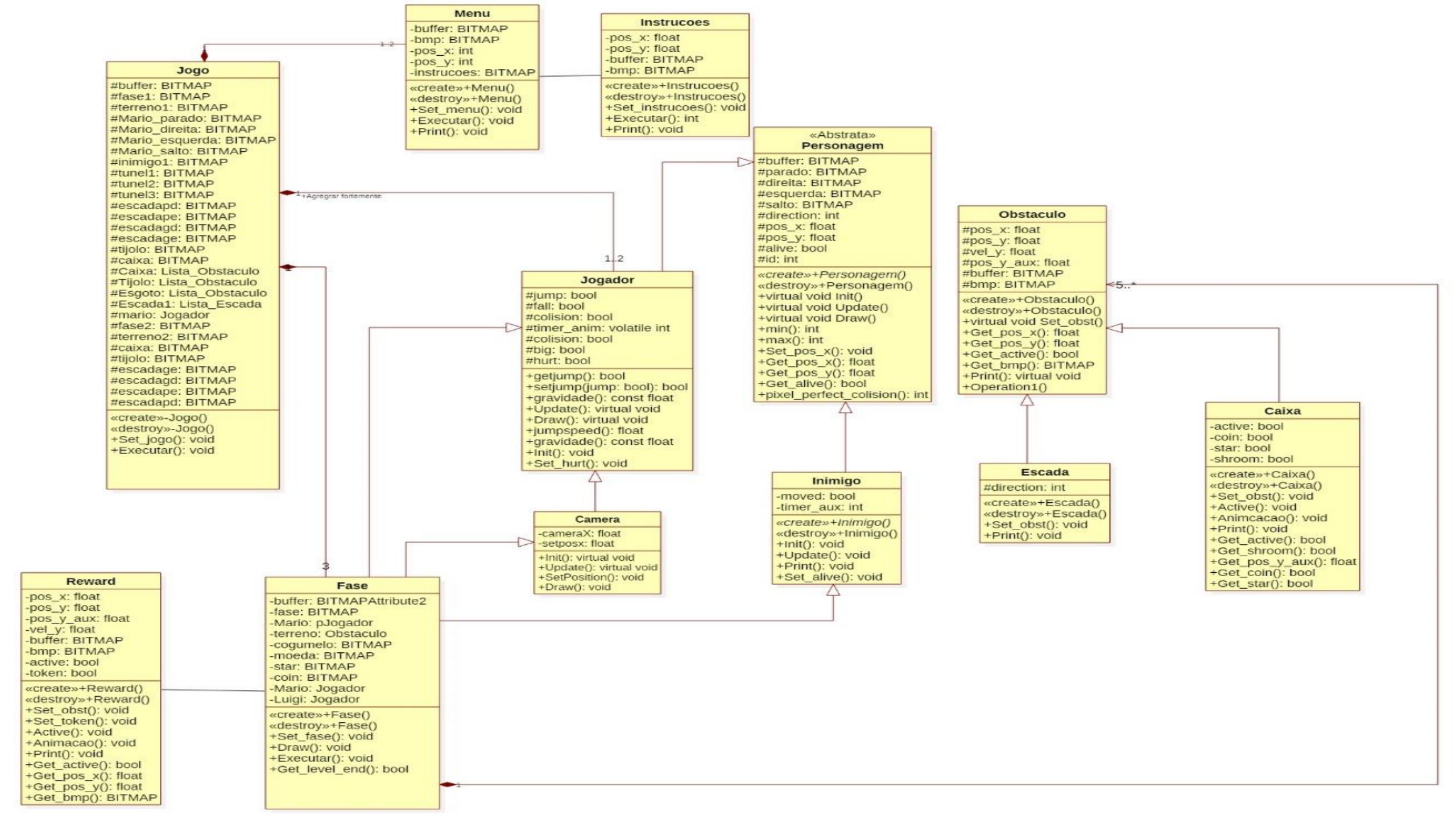


N.	Requisitos Funcionais	Situação
1	Apresentar menu de opções aos usuários do Jogo	Requisito previsto inicialmente e realizado
2	Permitir um ou dois jogadores aos usuários do Jogo	Requisito previsto inicialmente, mas não realizado
3	Disponibilizar ao menos três fases que podem ser jogadas sequencialmente ou selecionadas.	Requisito previsto inicialmente e realizado parcialmente – faltou ainda a última fase.
4	Ter seis tipos distintos de inimigos	Requisito previsto inicialmente e parcialmente realizado
5	Ter a cada fase ao menos dois tipos de inimigos com número aleatório de instâncias, podendo ser várias instâncias.	Requisito previsto inicialmente e realizado
6	Ter inimigo “Chefão” na última fase	Requisito previsto e não realizado
7	Ter quatro tipos de obstáculos.	Requisito previsto inicialmente e realizado
8	Ter em cada fase entre um e quatro tipos de obstáculos com número aleatório de obstáculos.	Requisito previsto inicialmente e realizado
9	Ter representação gráfica de instâncias.	Requisito previsto inicialmente e realizado
10	Ter em cada fase um cenário de jogo com obstáculos.	Requisito previsto inicialmente e realizado
12	Gerenciar colisões entre jogador e inimigos.	Requisito previsto inicialmente e realizado
13	Permitir cadastrar/salvar dados do usuário, manter pontuação durante jogo, salvar pontuação e gerar lista de pontuação (ranking).	Requisito previsto inicialmente, mas não realizado
14	Permitir Pausar o Jogo	Requisito previsto inicialmente, mas não realizado
15	Permitir Salvar Jogada.	Requisito previsto, mas não realizado

N.	Conceitos	Uso	Onde	
1	Elementares:			
	- Classes, objetos,	Sim	Todos .h e .cpp	
	- Atributos (privados), variáveis e constantes	Sim	Todos .h e .cpp	
	- Métodos (com e sem retorno).	Sim	Todos .h e .cpp	
	- Métodos (com retorno const e parâmetro const).	Sim	Todos .h e .cpp	
	- Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp	
	- Classe Principal.	Sim	Main.cpp & Principal.h/.cpp	
	- Divisão em .h e .cpp.	Sim	No projeto.	
2	Relações de:			
	- Associação	Sim	Jogador.h	
	- Agregação via associação	Sim		
	- Agregação propriamente dita.	Sim		
	- Herança elementar.	Sim		
	- Herança em diversos níveis.	Sim		
	- Herança múltipla.	SIm		
3	Ponteiros, generalizações e exceções			
	- Operador this	Sim	Método “Set” de todas as classes;	
	- Alocação de memória (new & delete)	Sim		
	- Gabaritos/Templates criada/adaptados pelos autores (e.g. Listas Encadeadas via Templates)	Não		
	- Uso de Tratamento de Exceções	Não		

4	Sobrecarga de:		
	- Construtoras e Métodos.	Não	
	- Operadores (2 tipos de operadores pelo menos)	Sim	
	Persistência de Objetos		
	- Texto via Arquivos de Fluxo	Não	
	- Binário	Não	
5	Virtualidade:		
	- Métodos Virtuais.	Sim	Obstaculo.h
	- Polimorfismo	Sim	Escada.h/Caixa.h
	- Métodos Virtuais Puros / Classes Abstratas	Sim	
	- Coesão e Desacoplamento	Sim	
6	Engenharia de Software		
	- Levantamento de Requisitos Textualmente e Tabelado (Ou por meio equivalente como Diagrama de Requisitos da SysML)	Sim	
	- Levantamento de Casos de Uso e sua expressão por meio de Diagrama de Casos de Uso em UML	Não	
	- Diagrama de Classes em UML	Sim	
	- Diagrama de Atividades em UML		
	- Outros diagramas em UML, Diag. de Estados, Diag. de Seqüência, Diag. de Pacotes etc. - E/ou outros diagramas estabelecidos, como Diag. de Fluxo de Dados (DFD) ou Diag. em SysML (Diag. de Requisitos,. de Blocos etc).	Não	
7	Biblioteca Gráfica		
	- Funcionalidades Elementares.	Sim	
	- Funcionalidades Avançadas como: • tratamento de colisões • duplo buffer • especificar aqui outras	Sim	Função de colisão em Personagem.h/Personagem.cpp Buffer: Método “Draw()” Camera.cpp
	Obs.: especificar quais funcionalidades.		
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc		
	- Ensino Médio (especificar quais Conceitos aqui)	Sim	Conceito de velocidade e aceleração Método Update de Jogador.cpp, linha 104.
	- Ensino Superior (especificar quais Conceitos aqui)	Não	

Organizadores:		
Espaço de Nomes (Namespace) criada pelos autores.	Não	
Classes aninhadas.	Sim	
Estáticos e String:		
Atributos estáticos e chamadas estáticas de métodos.	Não	
A classe Pré-definida String ou equivalente.	Não	
Standard Template Library (STL)		
Vector da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).		
List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).	Sim	Manipulação de obstáculos, Jogador.c, Fase.c, etc
Pilhas, Filas, Bifilas, Filas de Prioridade, Conjuntos, Multi-Conjuntos, Mapas ou Multi-Mapas*.	Sim	Idem
*Obs.: Listar apenas os utilizados		
Uso de Conceito Avançado no tocante a Orientação a Objetos.		
Ou Padrões de Projeto: GOF Ou Programação orientada a eventos e visual: Objetos gráficos como formulários, botões etc (Listar apenas os utilizados) Ou Programação concorrente: Threads (Linhas de Execução) no âmbito da Orientação a Objetos, utilizando Posix, C-Run-Time ou Win32API ou afins (com ou sem uso de Mutex, Semáforos, ou Troca de mensagens). Ou API de Comunicação em Rede: Cliente Servidor.	Não	


Menu
-buffer: BITMAP
-bmp: BITMAP
-pos_x: int
-pos_y: int
-instrucoes: BITMAP
«create»+Menu()
«destroy»+Menu()
+Set_menu(): void
+Executar(): void
+Print(): void
Instrucoes
-pos_x: float
-pos_y: float
-buffer: BITMAP
-bmp: BITMAP
«create»+Instrucoes()
«destroy»+Instrucoes()
+Set_instrucoes(): void
+Executar(): int
+Print(): void
Jogo
#buffer: BITMAP
#fase1: BITMAP
#terreno1: BITMAP
#Mario_parado: BITMAP
#Mario_direita: BITMAP
#Mario_esquerda: BITMAP
#Mario_salto: BITMAP
#inimigo1: BITMAP
#tunel1: BITMAP
#tunel2: BITMAP
#tunel3: BITMAP
#escadapd: BITMAP
#escadape: BITMAP
#escadagd: BITMAP
#escadage: BITMAP
#tijolo: BITMAP
#caixa: BITMAP
#Caixa: Lista_Obstaculo
#Tijolo: Lista_Obstaculo
#Esgoto: Lista_Obstaculo
#Escada1: Lista_Escada
#mario: Jogador
#fase2: BITMAP
#terreno2: BITMAP
#caixa: BITMAP
#tijolo: BITMAP
#escadage: BITMAP
#escadagd: BITMAP
#escadape: BITMAP
#escadapd: BITMAP
«create»-Jogo()
«destroy»-Jogo()
+Set_jogo(): void
+Executar(): void
«Abstrata»
Personagem
#buffer: BITMAP
#parado: BITMAP
#direita: BITMAP
#esquerda: BITMAP
#salto: BITMAP
#direction: int
#pos_x: float
#pos_y: float
#alive: bool
#id: int
«create»+Personagem()
«destroy»+Personagem()
+virtual void Init()
+virtual void Update()
+virtual void Draw()
+min(): int
+max(): int
+Set_pos_x(): void
+Get_pos_x(): float
+Get_pos_y(): float
+Get_alive(): bool
+pixel_perfect_colision(): int
Obstaculo
#pos_x: float
#pos_y: float
#vel_y: float
#pos_y_aux: float
#buffer: BITMAP
#bmp: BITMAP
«create»+Obstaculo()
«destroy»+Obstaculo()
+virtual void Set_obst()
+Get_pos_x(): float
+Get_pos_y(): float
+Get_active(): bool
+Get_bmp(): BITMAP
+Print(): virtual void
+Operation1()
Jogador
#jump: bool
#fall: bool
#colision: bool
#timer_anim: volatile int
#colision: bool
#big: bool
#hurt: bool
+getjump(): bool
+setjump(jump: bool): bool
+gravidade(): const float
+Update(): virtual void
+Draw(): virtual void
+jumpspeed(): float
+gravidade(): const float
+Init(): void
+Set_hurt(): void
Camera
-cameraX: float
-setposx: float
+Init(): virtual void
+Update(): virtual void
+SetPosition(): void
+Draw(): void
Inimigo
-moved: bool
-timer_aux: int
«create»+Inimigo()
«destroy»+Inimigo()
+Init(): void
+Update(): void
+Print(): void
+Set_alive(): void
Escada
#direction: int
«create»+Escada()
«destroy»+Escada()
+Set_obst(): void
+Print(): void
Caixa
-active: bool
-coin: bool
-star: bool
-shroom: bool
«create»+Caixa()
«destroy»+Caixa()
+Set_obst(): void
+Active(): void
+Animcacao(): void
+Print(): void
+Get_active(): bool
+Get_shroom(): bool
+Get_pos_y_aux(): float
+Get_coin(): bool
+Get_star(): bool
Reward
-pos_x: float
-pos_y: float
-pos_y_aux: float
-vel_y: float
-buffer: BITMAP
-bmp: BITMAP
-active: bool
-token: bool
«create»+Reward()
«destroy»+Reward()
+Set_obst(): void
+Set_token(): void
+Active(): void
+Animacao(): void
+Print(): void
+Get_active(): bool
+Get_pos_x(): float
+Get_pos_y(): float
+Get_bmp(): BITMAP
Fase
-buffer: BITMAPAttribute2
-fase: BITMAP
-Mario: pJogador
-terreno: Obstaculo
-cogumelo: BITMAP
-moeda: BITMAP
-star: BITMAP
-coin: BITMAP
-Mario: Jogador
-Luigi: Jogador
«create»+Fase()
«destroy»+Fase()
+Set_fase(): void
+Draw(): void
+Executar(): void
+Get_level_end(): bool
1..2
1
+Agregrar fortemente
1..2
1
3
5..*
1

Agradecimentos

- Prof Jean M.Simão