SUPER MARIO BROS REMAKE

Gorghe Teodor – 1211A

Facultatea de Automatică și Calculatoare – Iași

teodor.gorghe@student.tuiasi.ro

<2021-2022>

Despre jocul original:

Super Mario bros este un joc video de platformă dezvoltat și publicat de Nintendo și lansat în 1985 pe Nintendo Entertainment System . Acesta este primul joc din seria Super Mario. Acolo, jucătorul îl controlează pe Mario și călătorește prin Mushroom Kingdom pentru a o salva pe Prințesa Toadstool, care a fost capturată de inamicul principal, Bowser . Jocul se poate juca cu doi jucători, primul controlând Mario și al doilea Luigi , fratele acestuia din urmă.

Acest joc este unul dintre cele mai bine vândute jocuri din toate timpurile, cu peste 40 de milioane de exemplare vândute în întreaga lume. Ca joc de lansare, Super Mario Bros. a fost parțial responsabil pentru succesul sistemului Nintendo Entertainment, precum și pentru sfârșitul prăbușirii jocului video din 1983 din Statele Unite . Fiind unul dintre cele mai mari hituri obținute de Shigeru Miyamoto și Takashi Tezuka , jocul va da naștere la numeroase continuări și spinoff-uri.

Pe cont propriu, Super Mario Bros. a impus un design de nivel pentru toate jocurile care îl vor urma. Astfel, popularizează cu siguranță derularea orizontală , șefii și sub-șefii de la sfârșitul nivelului, comenzile rapide secrete , putând reporni jocul cu dificultăți crescute. Muzica jocului, compusă de Koji Kondo , este recunoscută la nivel mondial; de atunci a devenit reprezentativ pentru jocurile video în general.

Super Mario Bros este un joc de platformă în defilare orizontală împărțit în opt lumi fiecare cuprinzând patru niveluri. Jucătorul preia controlul lui Mario , care are abilitățile de a alerga și sări, printre altele, și se deplasează din partea stângă în partea dreaptă a ecranului pentru a ajunge la steagul care marchează sfârșitul fiecărui nivel - cu excepția ultimul nivel al fiecărei lumi, care are loc într-un castel cu un șef la sfârșit. Mario trece prin diferite tipuri de niveluri, de la medii în aer liber la niveluri subterane sau acvatice. Scopul ei este de a salva prințesa Toadstool din ghearele lui Bowser.

Este al doilea joc din seria de jocuri Mario, care dă naștere la un succes imens in această serie.

Despre jocul meu:

Jocul „Super Mario Bros Remake” a fost scris în C++, în care s-a folosit un framework-ul SDL2.

Proiectul cuprinde assets-urile din jocul original, Super Mario Bros, varianta NES, preluate de pe următoarele site-uri:

- https://www.spriters-resource.com/nes/supermariobros/
- https://themushroomkingdom.net/media/smb/wav

Jucătorul are următoarele abilități:

- de a se transforma din Small Mario la Super Mario, prin colectarea de mushroom.
- de a se transforma din Super Mario la Flower Mario, prin colectarea de flower.
- de a arunca cu flăcări
- de a sări
- de a turti Goomba
- de a pasa folosind Koopa Shell
- de a se teleporta prin Pipe
- de a sparge blocuri
- de a coborî steagul la sfârșit de nivel

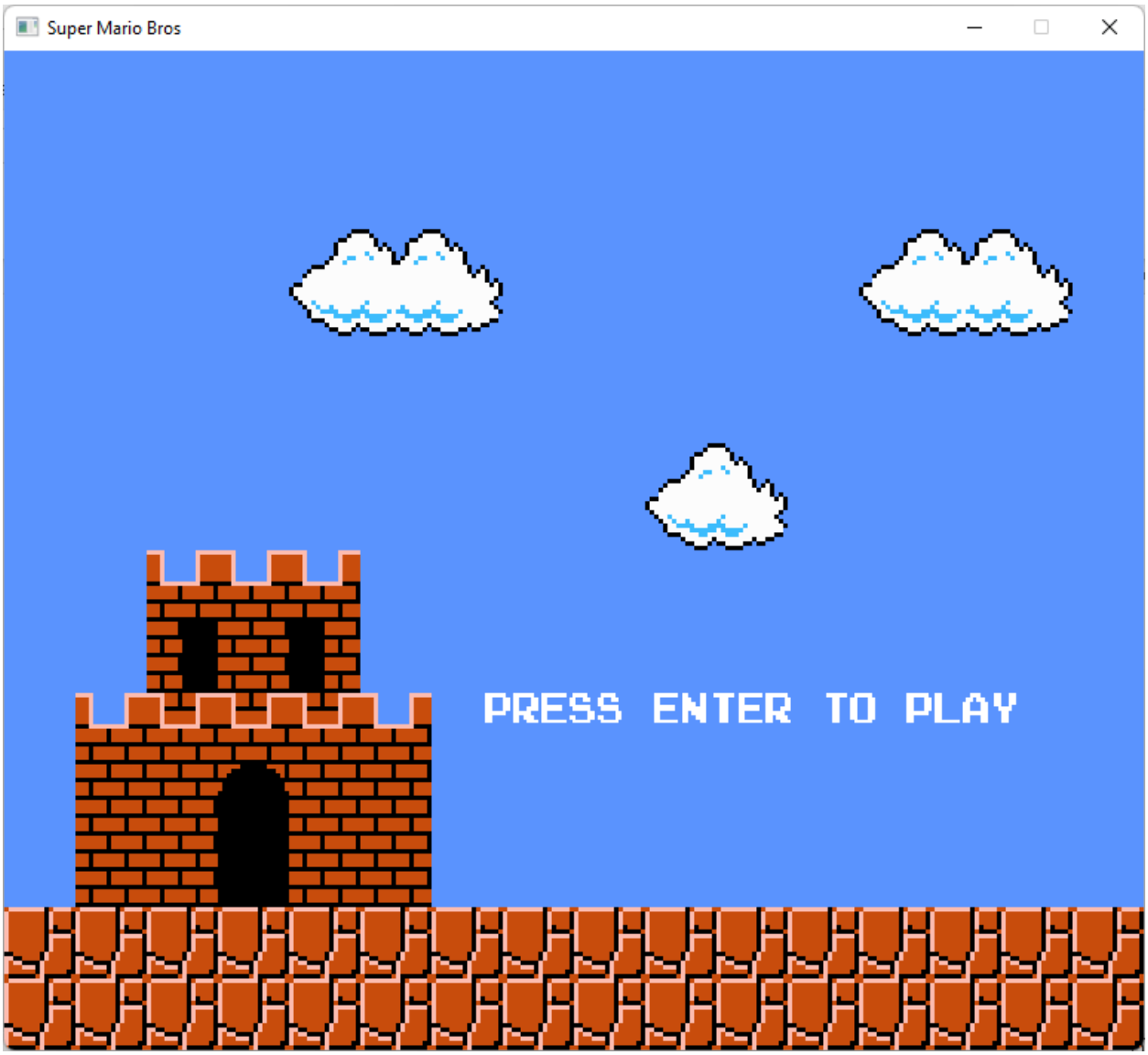
Față de varianta originală, ce mai este de lucru (foarte multe):

- construirea nivelelor până la 8-4
- construirea elementelor de nivel: platouri sus-jos, platouri cu scripete, Koopa Troopa roșu, bowser
- construirea temei UNDERWATER
- Bowser
- cutscene 1-2, etc...
- Cheep cheep
- remedieri animații (pipe, flower trow, flag, etc).

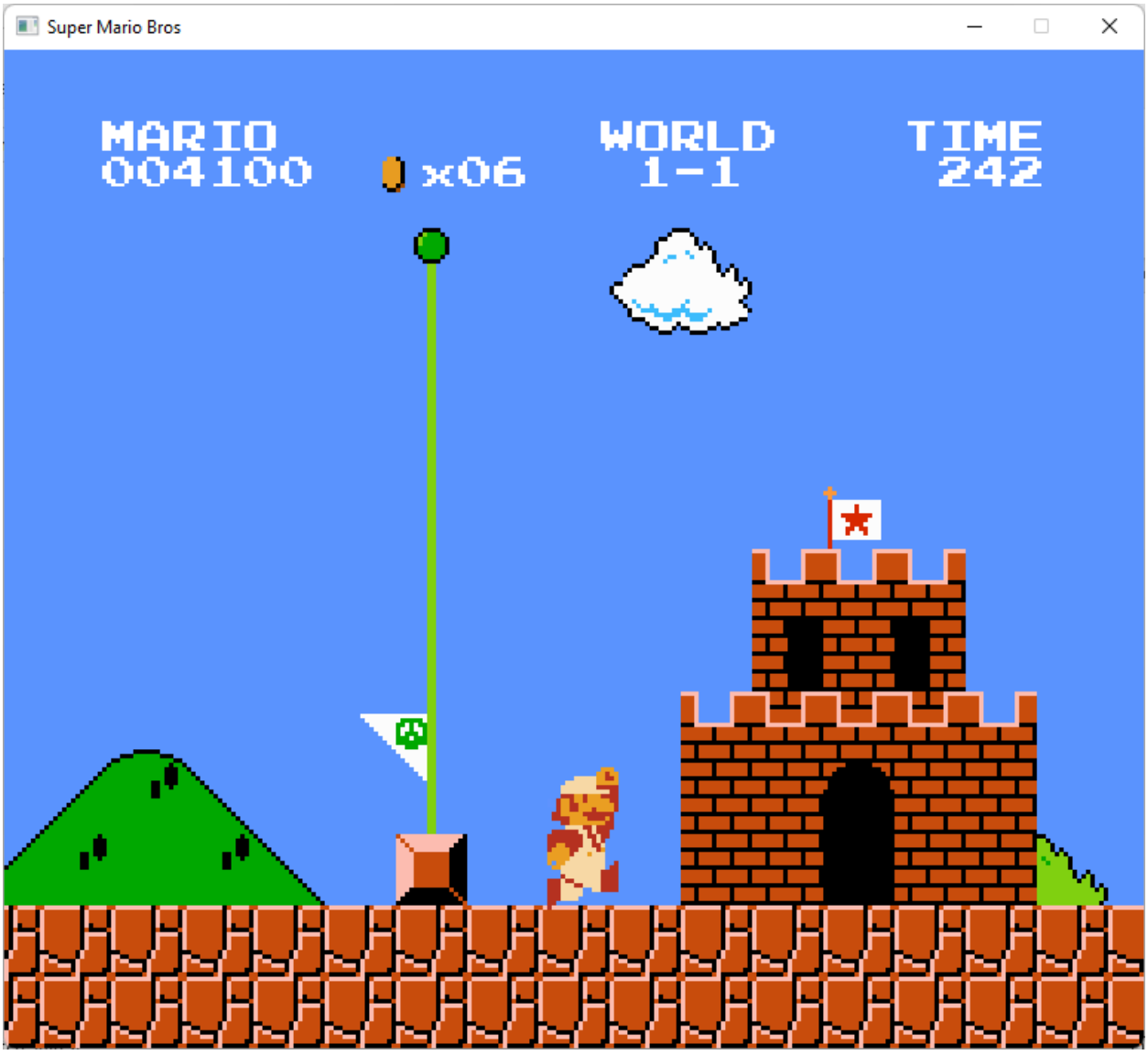
Pentru a rula acest joc, este necesar ca in directorul curent, să fie prezent toate librăriile dinamice din gama SDL2, SDL2_Image, SDL2_tff, SDL2_mixer.

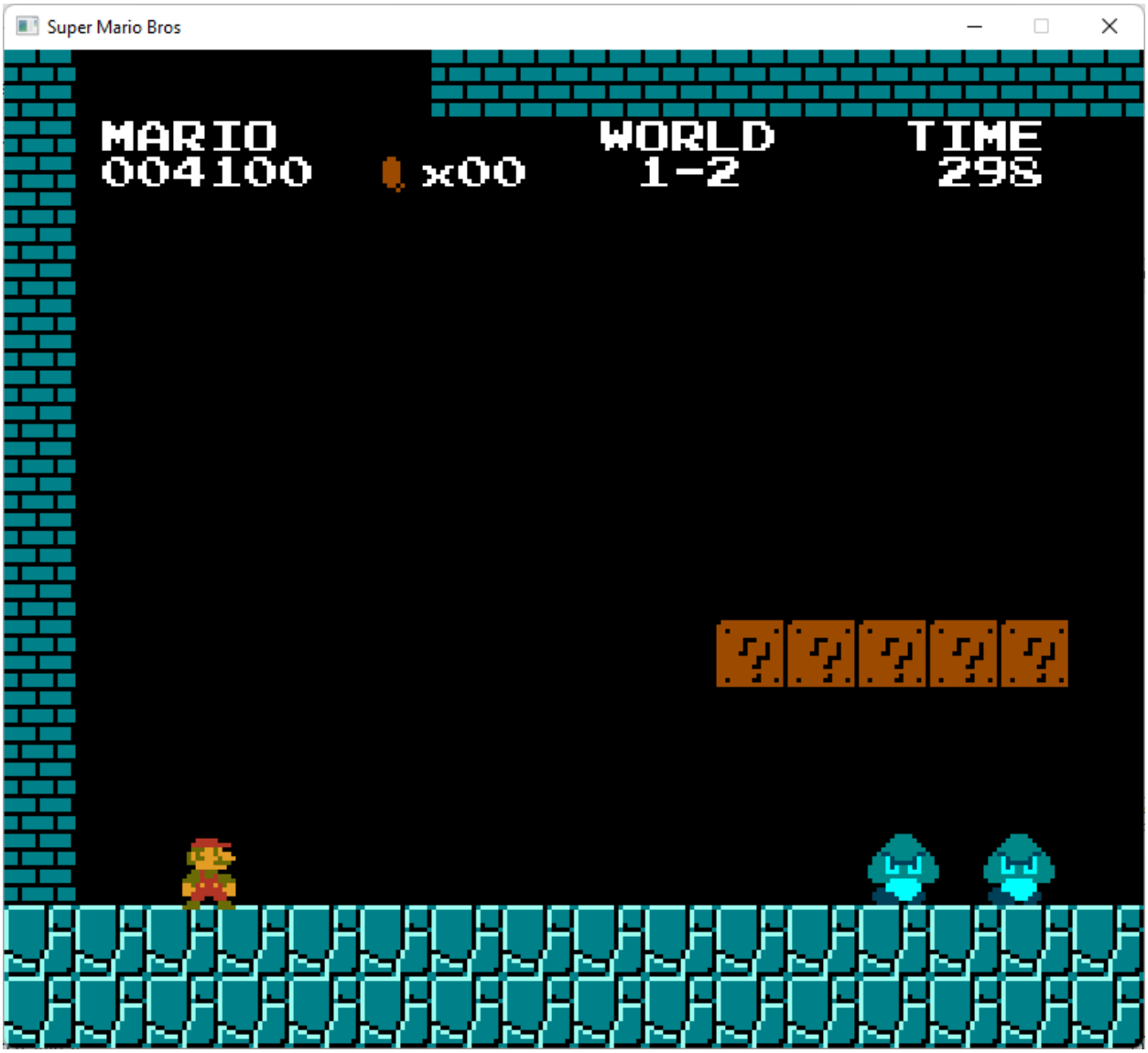
Cod proiect GIT: https://github.com/teodorgorghe/super-mario-remake

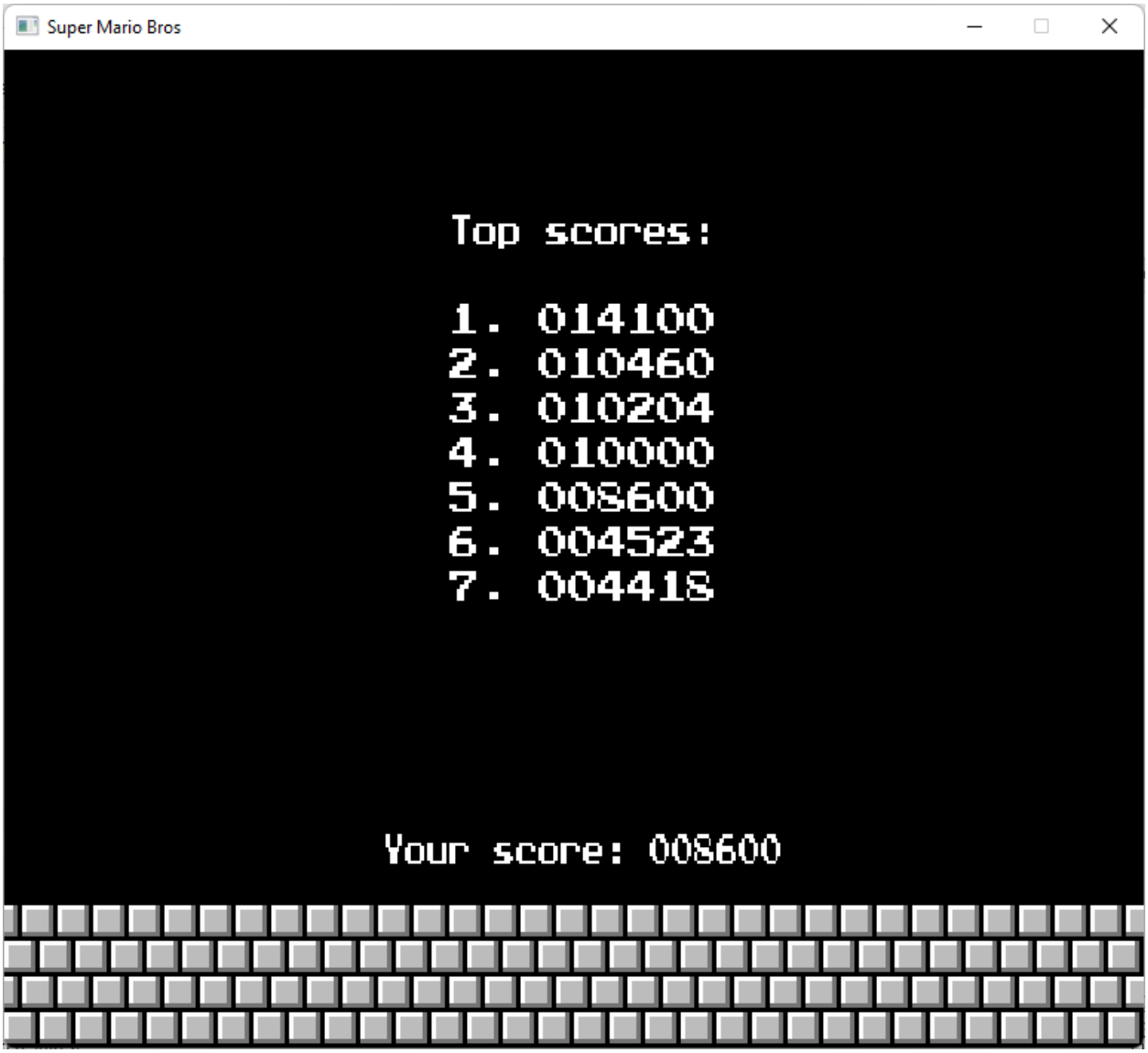
SCREENSHOTS:




Super Mario Bros
MARIO
000000
x00
WORLD
1-1
TIME
000
WORLD 1-1
x 3


Super Mario Bros
MARIO
004100
x06
WORLD
1-1
TIME
242


Super Mario Bros
MARIO
004100
x00
WORLD
1-2
TIME
298


Super Mario Bros
Top scores:
1. 014100
2. 010460
3. 010204
4. 010000
5. 008600
6. 004523
7. 004418
Your score: 008600

IMPLEMENTARE

Harta:

În primul rând, jocul are în prim-plan, citirea unui hărți, care are dimensiune implicită 240x14, salvată într-un fișier de tip .lvl, care are forma următoare:

[Lungime hartă][Lățime hartă] // Grid Hartă (reprezentat într-o structură de date, de tip TileType) [TileType(0,0):Textura, TexturaEditor, Flaguri][TileType(0,1)][TileType(0,2)]…. [TileType(0,13)][TileType(1,0)]…[TileType(239,13)]

Grid-ul se salvează pe o coadă de Tile-uri, urmând după aceea, în MapSystem, ca acele Tile-uri să fie create în world, cu anumite proprietăți.

În cazul în care se găsește un TileType, în care în Flags, are bitul corespunzător pentru Pipe, atunci, după secvența de TileType citită, se află și o secvență de date, în care se află datele despre Pipe (teleportare, directie pipe, etc...).

ENTITY-COMPONENT-SYSTEM

Pentru a facilita mai ușor crearea de caractere, Mario, Goomba, sau chiar a entităților din joc, cu diferite proprietăți la un moment dat, s-a preferat folosirea de sisteme, care să gestioneze mulțimea de entități, din clasa abstractă World.

Fiecare entitate are la rândul său, o mulțime de componente, care indică sistemelor ce să facă cu acea entitate.

De exemplu – Goomba:

- se atribuiesc următoarele componente:

- TextureComponent (pentru a sugera la RenderSystem să afișeze pe ecran caracterul)
- SolidComponent (pentru sistemul PhysicsSystem, care se ocupa cu coliziuni)
- KineticComponent (pentru a da mișcare caracterului - PhysicsComponent)
- EnemyComponent (pentru a gestiona toate interactiunile cu alti inamici/Mario)
- TransformComponent (pentru a da pozitie de pe harta – esentiala aproape de toate sistemele)
- WalkComponent (miscare a caracterului)
- GravityComponent (pentru a asigna caracterului forța gravitațională)

După anumite stări, Goomba mai poate avea anumite componente:

- CrushedComponent (pentru EnemySystem, dupa ce sare jucatorul pe Goomba, sa declanseze flag-ul de a-l face turtit)
- CallbackComponent (dupa un anumit timp, sa execute o comanda, de exemplu, sa dispara entitatea)
- etc.

Cum se instanțiază un jucător în lume:

```
auto player = world->create(); //Creaza o entitate noua, fara proprietati
player->assign<TransformComponent>(40, 140, TILE_SIZE - 4, SMALL_MARIO_HEIGHT);
//Assignare coordinate in plan + dimensiuni coliziuni.
player->assign<TextureComponent>(TextureId::MARIO_STAND); //Asignare textura Mario
player->assign<SolidComponent>(); //Flag activare coliziuni
player->assign<PlayerComponent>(); //Asignare miscare + abilitati jucator (control, etc).
player->assign<GravityComponent>(); //Asignare gravitate (este necesara si in cazul UNDERWATER)
//player->assign<SwimComponent>(); // Cand Mario este sub apa.
player->assign<KineticComponent>(); //Asignare miscare jucator, care va fi gestionata de PlayerSystem
```

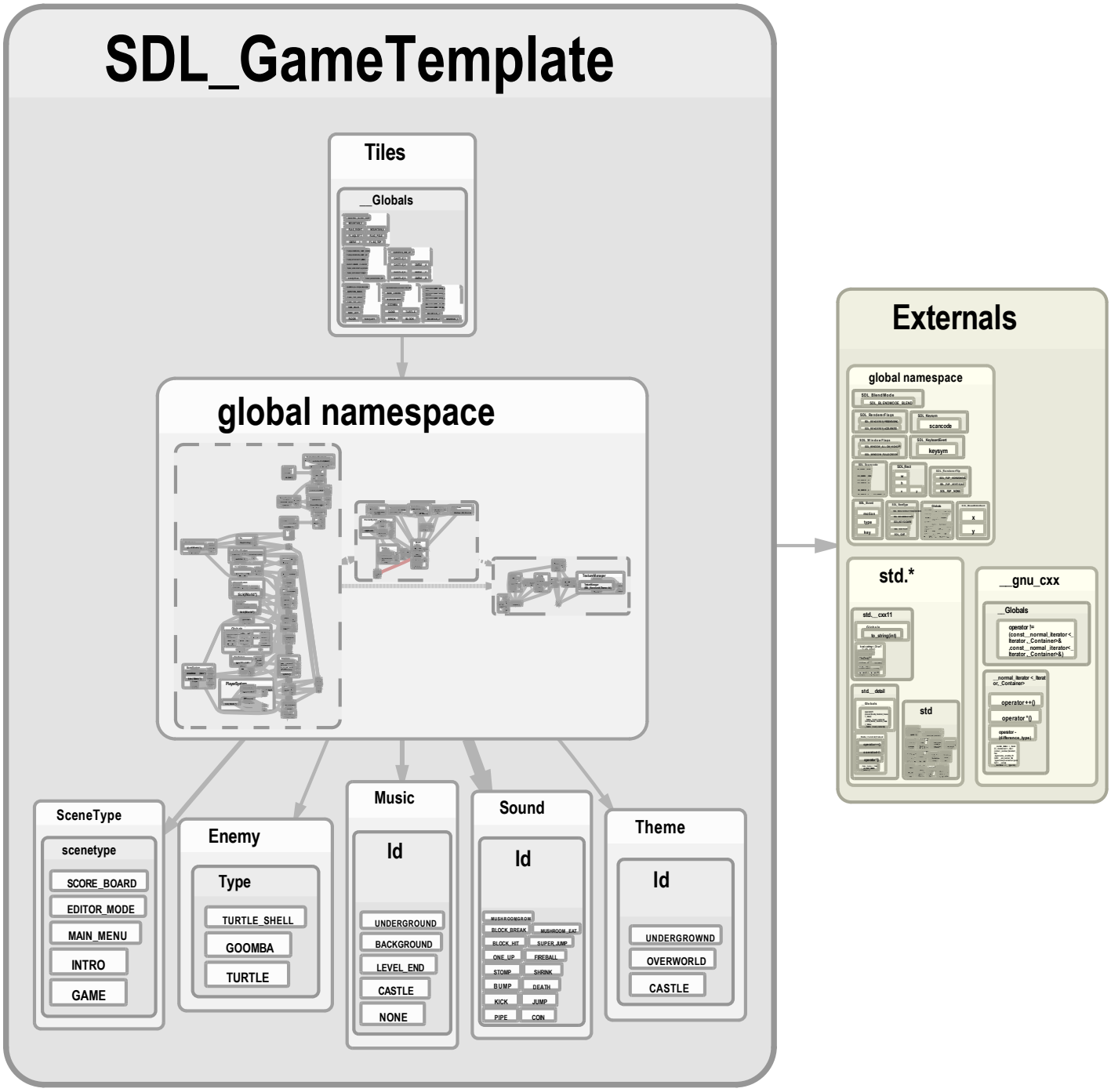
Cum se sterge un jucator:

```
player->clearComponents();
```

Marea majoritatea componentelor au rol de flag, dar pot transporta date printre celelalte sisteme.

Accesarea unei entitati dintr-un alt sistem se face prin cautarea entitatilor care au cateva componente specifice:

```
world->find<[componenta necesara…]>([functia care se apeleaza cand gaseste acea componenta]);
// Se comporta ca un for intre componentele gasite.
```


SDL_GameTemplate
Tiles
__Globals
global namespace
Externals
global namespace
SDL_BlendMode
SDL_BLENDMODE_BLEND
SDL_RendererFlags
SDL_RENDERER_SOFTWARE
SDL_RENDERER_ACCELERATED
SDL_RENDERER_PRESENTVSYNC
SDL_RENDERER_TARGETTEXTURE
SDL_WindowFlags
SDL_WINDOW_ALLOW_HIGHDPI
SDL_WINDOW_MAXIMIZED
SDL_WINDOW_MINIMIZED
SDL_WINDOW_FULLSCREEN
SDL_WINDOW_FULLSCREEN_DESKTOP
SDL_WINDOW_OPENGL
SDL_WINDOW_RESIZABLE
SDL_WINDOW_SHOWN
SDL_WINDOW_HIDDEN
SDL_WINDOW_BORDERLESS
SDL_WINDOW_INPUT_GRABBED
SDL_WINDOW_INPUT_FOCUS
SDL_WINDOW_MOUSE_FOCUS
SDL_WINDOW_FOREIGN
SDL_WINDOW_ALLOW_HIGHDPI
SDL_Keycode
scancode
SDL_KeyboardEvent
keysym
SDL_Scancode
SDL_Rect
x
y
w
h
SDL_RendererFlip
SDL_FLIP_NONE
SDL_Event
motion
type
key
Globals
x
y
std.*
std.__cxx11
to_string[abi]
std.__detail
std
__gnu_cxx
__Globals
operator != (const __normal_iterator<_Iterator,_Container>& ,const __normal_iterator<_Iterator,_Container>&)
__normal_iterator<_Iterator,_Container>
operator++()
operator*()
operator-(difference_type)
SceneType
scenetype
SCORE_BOARD
EDITOR_MODE
MAIN_MENU
INTRO
GAME
Enemy
Type
TURTLE_SHELL
GOOMBA
TURTLE
Music
Id
UNDERGROUND
BACKGROUND
LEVEL_END
CASTLE
NONE
Sound
Id
MUSHROOMGROW
BLOCK_BREAK
MUSHROOM_EAT
BLOCK_HIT
SUPER_JUMP
ONE_UP
FIREBALL
STOMP
SHRINK
BUMP
DEATH
KICK
JUMP
PIPE
COIN
Theme
Id
UNDERGROWND
OVERWORLD
CASTLE

BAREM DE NOTARE:

#	Criteriu	Realizat
1	Abstractizare	DA
2	Încapsulare	DA
3	Moștenire (ierarhie de grad 3 minim)	DA
4	Polimorfism	DA
5	Interfețe (clase abstracte)	DA
6	Gestionarea erorilor (excepții)	DA
7	Salvarea sau încărcarea configurației jocului (Lucrul cu fișiere)	DA
8	Număr de niveluri cu dificultate graduală (minim 3)	DA