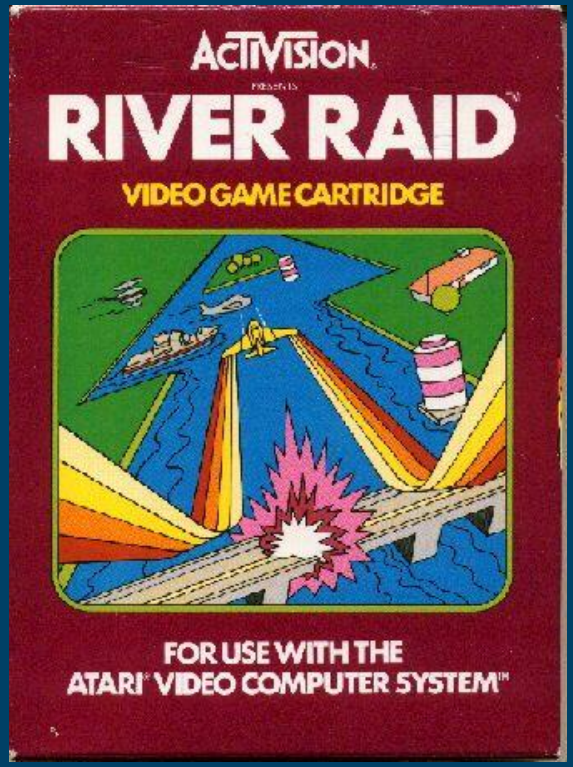
Beach Raid

Projeto Final Reativos

Camila Gusmão e Renato Domingues

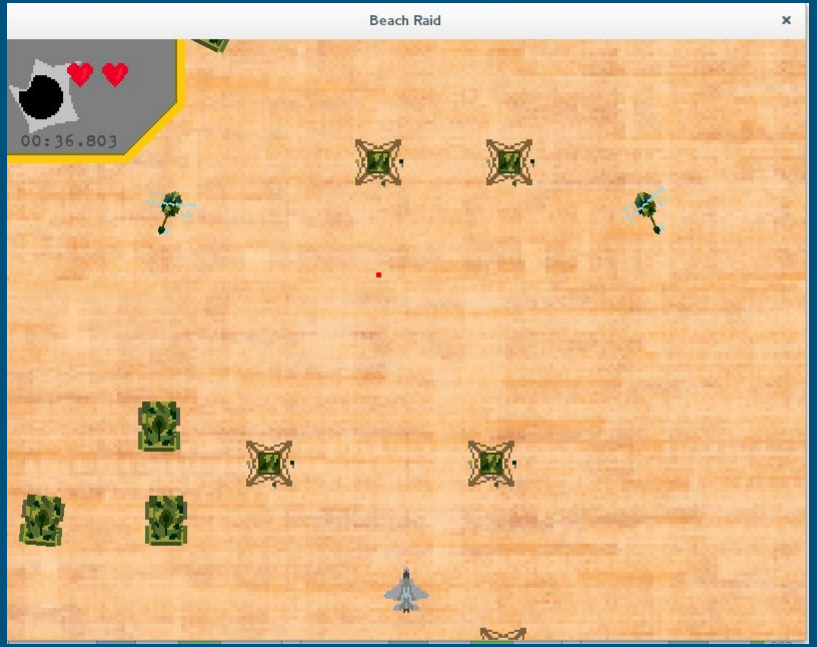
Ideia do Projeto

River Raid

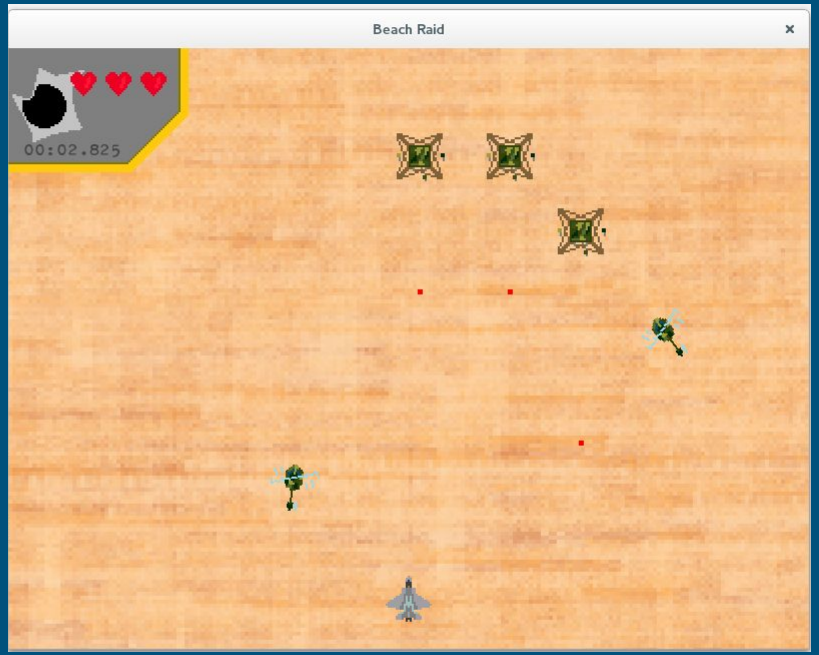


Beach Raid

- O jogador controla uma nave que possui combustível e munição ilimitados;
- Existem 3 classes de inimigos:
  - Torre: Atiram e não se movem;
  - Tanque: Atiram e se locomovem na diagonal;
  - Helicóptero: Não atiram e realizam trajetória senoidal.



Beach Raid

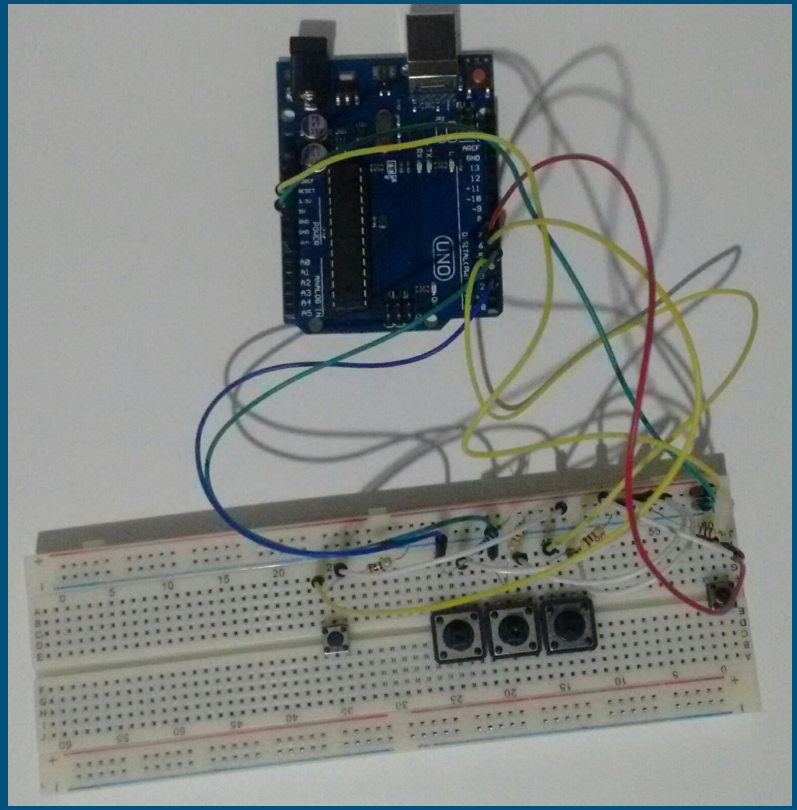


- O jogo possui 3 níveis de dificuldade:
  - FÁCIL: 3 vidas, menor qtde de inimigos;
  - MÉDIO: 3 vidas;
  - DIFÍCIL: 1 vida.
- A nave perde uma vida se for atingida por um tiro ou colidir com um inimigo;
- O objetivo é sobreviver o máximo de tempo possível.

Controlador Arduino

Componentes

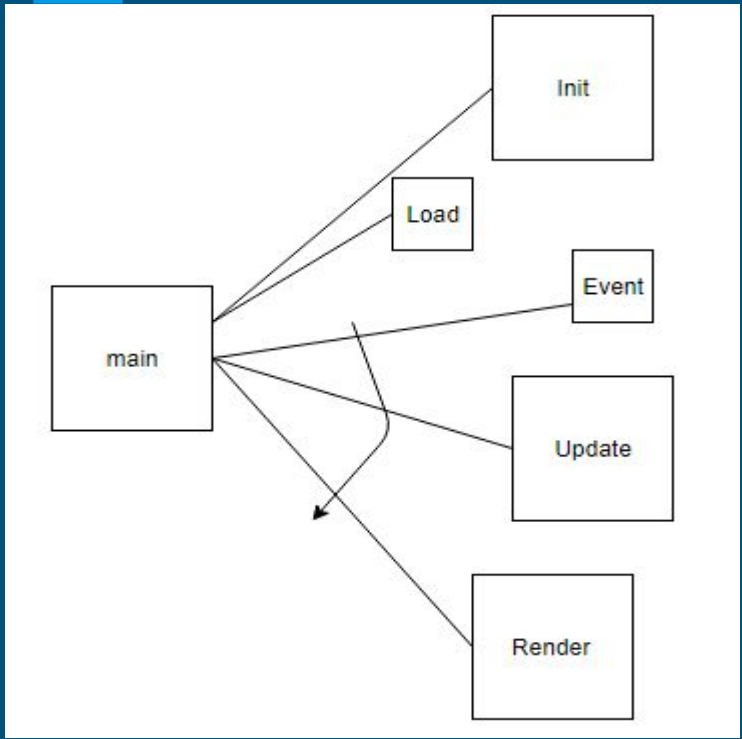
- Arduino Uno
- Protoboard 830 pinos
- 5 botões
- Fios macho-macho
- Resistores

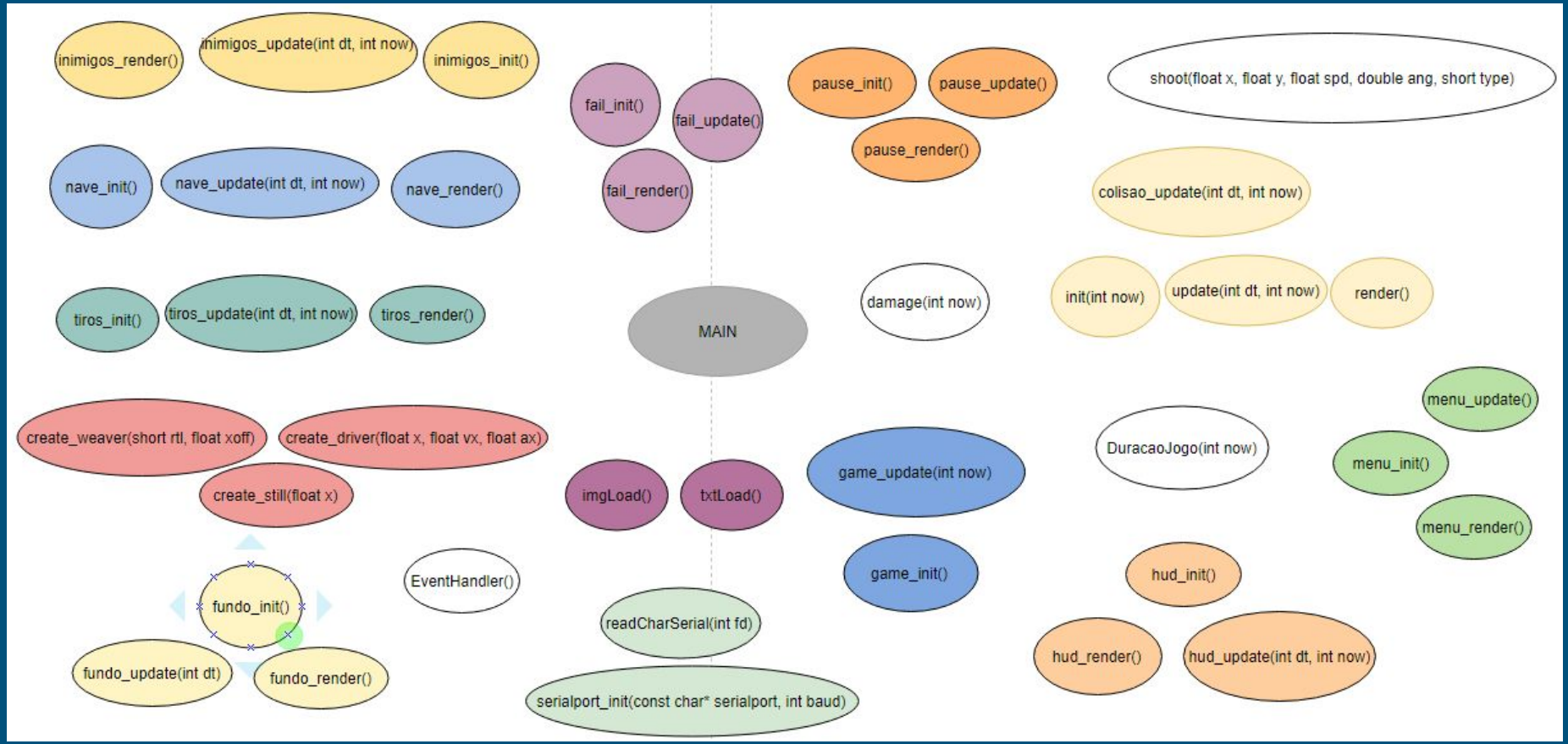


Comunicação Arduino via serial:
https://github.com/todbot/arduino-serial

{Código}

Estrutura




inimigos_render()
inimigos_update(int dt, int now)
inimigos_init()
fail_init()
fail_update()
fail_render()
pause_init()
pause_update()
pause_render()
shoot(float x, float y, float spd, double ang, short type)
nave_init()
nave_update(int dt, int now)
nave_render()
colisao_update(int dt, int now)
tiros_init()
tiros_update(int dt, int now)
tiros_render()
MAIN
damage(int now)
init(int now)
update(int dt, int now)
render()
menu_update()
create_weaver(short rtl, float xoff)
create_driver(float x, float vx, float ax)
create_still(float x)
DuracaoJogo(int now)
menu_init()
menu_render()
imgLoad()
txtLoad()
game_update(int now)
fundo_init()
EventHandler()
game_init()
hud_init()
readCharSerial(int fd)
fundo_update(int dt)
fundo_render()
hud_render()
hud_update(int dt, int now)
serialport_init(const char* serialport, int baud)

```c
void init(int now){
    switch(_gamestate){
        case SMENU:
        ispause=0;
        menu_init();
        break;
        case SFAIL:
        fail_init();
        break;
        case SPAUSE:
        ispause=1;
        pause_init();
        pause_start = now;
        break;
        case SGAME:
        if(!ispause){
            tempo_inicial = now;
            nave_init();
            tiros_init();
            inimigos_init();
            game_init();
            hud_init();
            fundo_init();
        }else{
            tempo_inicial+=now-pause_start;
        }
        ispause=0;
        break;
    }
    //jogo começou
}
```

```c
// game logic
void update(int dt,int now){
    switch(_gamestate){
        case SMENU:
        menu_update();
        break;
        case SFAIL:
        fail_update();
        break;
        case SPAUSE:
        pause_update();
        break;
        case SGAME: //jogando
        nave_update(dt,now);
        tiros_update(dt,now);
        inimigos_update(dt,now);
        colisao_update(now);
        game_update(now);
        fundo_update(dt);
        hud_update(dt,now);
    }
    if(gamestate!=_gamestate){
        _gamestate=gamestate;
        init(now);
    }
    for(i=0;i<BTNQTD;i++){
        keyst[i] = keyb[i];
    }
}
```

```c
void render(){
    // fps calculation
    if ((now-start) > 1000) {
        printf("FPS = %d\n", frames);
        frames = 0;
        start = now;
    }
    frames++;

    // clear all
    SDL_SetRenderDrawColor(ren,255,255,255,0xFF);
    SDL_RenderFillRect(ren, &black);

    // render entities
    switch(_gamestate){
        case SMENU:
        menu_render();
        break;
        case SFAIL:
        fail_render();
        break;
        case SPAUSE:
        pause_render();
        break;
        case SGAME:
        fundo_render();
        tiros_render();
        inimigos_render();
        nave_render();
        hud_render();
        break;
    }
    // update screen
    SDL_RenderPresent(ren);
}
```

```c
void colisao_update(int now){
    short ie, it;
    for(ie=0;ie<50;ie++){
        if(inimigo[ie].on){
            for(it=0;it<50;it++){// enemy-tiros colision
                if(tiros[it].on && tiros[it].type==0){
                    SDL_bool col = SDL_HasIntersection(&inimigo[ie].r,&tiros[it].r);
                    if(col){// Bang! You killed someone
                        tiros[it].on=0;
                        inimigo[ie].on=0;
                    }
                }
            }
            if(inimigo[ie].type<2){// enemy-player colision
                SDL_bool col = SDL_HasIntersection(&inimigo[ie].r,&nave.r);
                if(col){// Bang *dead*!
                    inimigo[ie].on=0;
                    damage(now);
                }
            }
        }
    }
    for(it=0;it<50;it++){// tiros-player colision
        if(tiros[it].on && tiros[it].type==1){
            SDL_bool col = SDL_HasIntersection(&tiros[it].r,&nave.r);
            if(col){// Bang *dead*!
                tiros[it].on=0;
                damage(now);
            }
        }
    }
}
```