UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERÍA Y PRODUCCIÓN DE SERVICIOS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

"AÑO DEL BICENTENARIO DEL PERÚ: 200 AÑOS DE INDEPENDENCIA"

CÁLCULO EN VARIAS VARIABLES

GRUPO: A SEMESTRE: 2021 - A

PROYECTO DE UN VIDEOJUEGO: COVID MAN

INTEGRANTES:

Jheeremy Manuel, Alvarez Astete

Ronald Romario, Gutierrez Arratia

Diego Raul, Huanca Rivas

Josue Gabriel, Sumare Uscca

Yanira Angie, Suni Quispe

DOCENTE:

Alvaro Henry Mamani Aliaga

Arequipa – Perú

2021

ÍNDICE

1. Introducción

Covid-Man es un juego interactivo, inspirado en la temática del juego de los 80 's Pacman y basado en el contexto actual es decir en el Covid-19.

En esta ocasión pacman es un doctor que tiene que obtener una cantidad de puntos para pasar de nivel, este podrá obtener puntos al recolectar pastillas y vacunas contra el covid-19 que se encuentran esparcidas por el mapa, además se tiene como enemigo al covid-19 , durante todo el juego el doctor deberá protegerse del virus sin tocarlo de lo contrario perderá una vida, hasta morir.

Se hizo puso en práctica temas tocados en el curso , mediante el uso de gestión dinámica de memoria en la creación de objetos , la creación de clases se realizó mediante el patrón Factory method, mediante este patrón se pudo realizar herencia , al aplicar herencia pudimos aplicar también el polimorfismo virtual en estas clases hijas o concretas.

2. Objetivos

- Poner en práctica nuestros conocimientos en el lenguaje C++, y herramientas que nos puedan ayudar a programar.
- Demostrar la capacidad de abstracción de clases llevándolo a un entorno más interactivo y aplicativo como lo es un videojuego.
- Familiarizarnos con el uso de interfaces gráfica**s**

3. Herramientas

- Lenguaje de programación C++.
- Code Blocks o Dev C++.
- Librería para programación de juegos Allegro (Atari Low Level Game Routines).

4. Desarrollo de Software

Buffer.h

```
#define MAXFILAS 20 //Para el eje y (19)
#define MAXCOLS  31 //Para el eje x (30)
char mapa1[MAXFILAS][MAXCOLS]={
        "XXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
        "XooooooooooXXXXXooooooooooooo",
        "X XXX XXXXX XXXXX XXXXX XXXoX",
        "X XXX XXXXX XXXXX XXXXX XXXoX",
        "X ooooooooooooooooooooooooooX",
        "X XXX XX XXXXXXXXXX XX XXXoX",
        "X vvv XXoooooXXXooooooXX    oX",
        "X XXX XXXXXX XXX XXXXXXoXXXoX",
        "XXXXX XX             XX XXXXX",
        "oooooXX XXXXXXXXXXX XXooooooo",
        "X XXX XX XXXXXXXXXX XX XXX X",
        "X XXX XXooooooooooooXX XXX X",
        "X XXX XXXXXX XXX XXXXXX XXX X",
        "XoooooXXooooooXXXooooooXXooooX",
        "X XXX XX XXXXXXXXXX XX XXXoX",
        "X XXXooooooooooooooooooXXXoX",
        "X XXX XXXX XXXXXXXX XXX XXX X",
        "X XXX XXXX          XXX XXX X",
        " ooooooooooXXXXXXXXooooooooooX",
        "XXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    };
char mapa2[MAXFILAS][MAXCOLS]={
        "XXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
        "X  vvvv     XXXXX    vvvv    ",
        "X XXX XXXXX XXXXX XXXXX XXX X",
        "X XXX    XX   X  XX     XXX X",
        "X   vvv    o  vvv           X",
        "X XXX XX XXXXXXXXXX XX XXX X",
        "X     XX  o  XXX     XX     X",
        "X XXX XXXXXX XXX XXXXXX X X X",
        "X XXX XX  vv   o     XX X X X",
        "       XX XXXXXXXXXX XX      ",
        "X XXX XX XXXX   XXXX XX XXXXX",
        "X XXX XX         vv  XX XXX  ",
        "X XXX XXXXXX XXX XXXXXX XXX X",
        "X     XX     XXX     XX     X",
        "X XXX XX XXXXXXXXXX XX XXX X",
        "X XXX  o  vv   o   vv   XXX X",
        "X XXX XXXX XXXXXXXX XXX XXX X",
        "X XXX XXXX          XXX XXX X",
        "    vvv    XXXXXXXX   vvv    ",
        "XXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
```

```
    };
struct Buffer{
    BITMAP *buffer;
    int ejeX=880;
    int ejeY=700;
}buffer;
```

Obstaculos.h

```
class Obstaculos{
    private:
    public:
    virtual int getEjeX()=0;
    virtual int getEjeY()=0;
};
```

Casa.h

```
#include "Obstaculos.h"
#include <allegro.h>
class Casa : public Obstaculos{
   private:
   BITMAP *casa=NULL;//Variable de la imagen del muro;
   const char** tipoCasa;
   int ejeX;
   int ejeY;
   int tipo;
   public:
   Casa();
   ~Casa();
   int getEjeX();
   int getEjeY();
   void setTipo(int);
   void definirCasa();
   BITMAP* getCasa();
};
```

Casa.cpp

```
#include "Casa.h"
Casa::~Casa(){
   delete []tipoCasa;
}
Casa::Casa(){
   //this->casa=load_bitmap("casita.bmp",nullptr);//
   tipoCasa=new const char*[3];
   tipoCasa[0]="casita.bmp";
   tipoCasa[1]="casa1.bmp";
   tipoCasa[2]="casa2.bmp";
   this->ejeX=30;
   this->ejeY=30;
}
int Casa::getEjeX(){
   return ejeX;
}
int Casa::getEjeY(){
   return ejeY;
}
BITMAP* Casa::getCasa(){
   return casa;
}
void Casa::setTipo(int tipo){
   this->tipo=tipo;
}
void Casa::definirCasa(){
```

```
  if(tipo%2==0){
    this->casa=load_bitmap("casita.bmp",NULL);
  }
  else if(tipo%3==0){
    this->casa=load_bitmap("casa1.bmp",NULL);
  }
  else{
    this->casa=load_bitmap("casa2.bmp",NULL);
  }
}
```

Comida.h

```
class Comida{
    private:
    public:
    virtual int getEjeX()=0;
    virtual int getEjeY()=0;
};
```

Pastilla.h

```
#include "Comida.h"
class Pastilla : public Comida{
   private:
   BITMAP *pastilla=NULL;//Variable de la imagen del muro;
   int ejeX;
   int ejeY;
   public:
   Pastilla();
   ~Pastilla();
   int getEjeX();
   int getEjeY();
   BITMAP* getPastilla();
};
```

Pastilla.cpp

```
#include "Pastilla.h"
Pastilla::~Pastilla(){

}
Pastilla::Pastilla(){
   this->pastilla=load_bitmap("pastilla.bmp",NULL);//
   this->ejeX=30;
   this->ejeY=30;
}
int Pastilla::getEjeX(){
   return ejeX;
}
int Pastilla::getEjeY(){
   return ejeY;
}
BITMAP* Pastilla::getPastilla(){
   return pastilla;
}
```

Vacuna.h

```
#include "Comida.h"
class Vacuna : public Comida{
   private:
   BITMAP *vacuna=NULL;//Variable de la imagen del muro;
   int ejeX;
   int ejeY;
   public:
   Vacuna();
   ~Vacuna();
   int getEjeX();
   int getEjeY();
   BITMAP* getVacuna();
};
```

vacuna.cpp

```
#include "Vacuna.h"
Vacuna::~Vacuna(){

}
Vacuna::Vacuna(){
   this->vacuna=load_bitmap("vacuna.bmp",NULL);//
   this->ejeX=30;
   this->ejeY=30;
}
int Vacuna::getEjeX(){
   return ejeX;
}
int Vacuna::getEjeY(){
   return ejeY;
}
BITMAP* Vacuna::getVacuna(){
   return vacuna;
}
```

personaje.h

```
#include "Buffer.h"
class Personaje{
     public:
     virtual int getPosX()=0;
     virtual int getPosY()=0;
     virtual void setNumMapa(int)=0;
     virtual void movimiento()=0;
     virtual void dibujar()=0;
     virtual void atajo(int)=0;
};
```

Covid_man.h

```
#include <allegro.h>
#include "Enfermo.cpp"
class Covid_man : public Personaje{
    private:
    SAMPLE *muerte=NULL;
```

```
    BITMAP *pacbmp=NULL;//Creara un espacio en donde estara pacman
    BITMAP *muertebmp=NULL;
    BITMAP *pacman=NULL;//Dimensiones de un pacman;//Imagen pacman
    int dir;//para que pacman no se mueva al iniciar el juego
    int px,py;//posicion adecuada de pacman
    int numMapa;
    int vidas;
    public:
    Covid_man();
    ~Covid_man();

    SAMPLE* getMusicMuerte();
    BITMAP* getMuertebmp();
    void setPacman(BITMAP* pacman);
    BITMAP* getPacman();

    int getVida();
    const char* imprimirVida(int);
    void setVida(int);
    int getPosX();//
    int getPosY();//
    void setNumMapa(int);//
    void movimiento();//
    void dibujar();//
    void morderPacman();
    void inicioMuerte();
    void finalMuerte(int);
    void atajo(int);//
    void ganarVida(int puntos);

};
```

Covid_man.cpp

```
#include "Covid_man.h"
SAMPLE* Covid_man::getMusicMuerte(){
    return muerte;
}
Covid_man::~Covid_man(){

}
Covid_man::Covid_man(){
    this->muerte= load_wav("muerte.wav");
    this->pacbmp=load_bitmap("covid_man.bmp",NULL);//Crea espacio donde estará pacman
    this->muertebmp=load_bitmap("muerte.bmp",NULL);
    this->pacman=create_bitmap(33,33);//Dimensiones de un pacman;//Imagen pacman
    this->dir=4;//para que pacman no se mueva al iniciar el juego
    this->px=30*14;
    this->py=30*17;//posicion adecuada de pacman
    this->numMapa=1;
    this->vidas=3;
}
BITMAP* Covid_man::getMuertebmp(){
    return muertebmp;
    }
void Covid_man::setPacman(BITMAP* pacman){
    this->pacman=pacman;
    }
```

```
BITMAP* Covid_man::getPacman(){
    return this->pacman;
    }
void Covid_man::inicioMuerte(){
    clear(pacman);
    clear(buffer.buffer);
}
void Covid_man::finalMuerte(int i){
    blit(muertebmp,pacman,i*33,0,0,0,33,33);
    draw_sprite(buffer.buffer,pacman,px,py);
}
const char* Covid_man::imprimirVida(int vida){
    if(vida==0)
        return "0";

    else if(vida==1)
        return "1";

    else if(vida ==2)
        return "2";

    else if(vida ==3)
        return "3";

    else if(vida ==4)
        return "4";

    else if(vida ==5)
        return "5";

    else if(vida ==6)
        return "6";

    return "error";
}
void Covid_man::movimiento(){
    if(key[KEY_LEFT])
            dir=0;
    else if(key[KEY_RIGHT])
            dir=1;
    else if(key[KEY_UP])
            dir=2;
    else if(key[KEY_DOWN])
            dir=3;

    if(numMapa==1){//Nivel 1 del juego
        if(dir==0){
          if(mapa1[py/30][(px-30)/30]!='X')//Evitamos que pacman cruce los muros
             px=px-30;
          else
             dir=4;
        }
        else if(dir==1){
           if(mapa1[py/30][(px+30)/30]!='X')
             px=px+30;
           else
             dir=4;
        }
```

```
        else if(dir==2){
           if(mapa1[(py-30)/30][(px)/30]!='X')
             py=py-30;
           else
             dir=4;
         }

        else if(dir==3){
           if(mapa1[(py+30)/30][(px)/30]!='X')
              py=py+30;
           else
              dir=4;
        }
    }
    else if(numMapa==2){//Nivel 2 del juego
        if(dir==0){
          if(mapa2[py/30][(px-30)/30]!='X')//Evitamos que pacman cruce los muros
             px=px-30;
          else
             dir=4;
        }
        else if(dir==1){
           if(mapa2[py/30][(px+30)/30]!='X')
             px=px+30;
           else
             dir=4;
        }
        else if(dir==2){
           if(mapa2[(py-30)/30][(px)/30]!='X')
             py=py-30;
           else
             dir=4;
         }

        else if(dir==3){
           if(mapa2[(py+30)/30][(px)/30]!='X')
              py=py+30;
           else
              dir=4;
        }
    }
    Covid_man::atajo(numMapa);
    clear(buffer.buffer);//borramos posiciones anteriores de pacman
}
void Covid_man::dibujar(){

    blit(pacbmp,pacman,dir*33,0,0,0,33,33);
    draw_sprite(buffer.buffer,pacman,px,py);//para respetar transparencia

}
void Covid_man::morderPacman(){
    clear(pacman);
    blit(pacbmp,pacman,4*33,0,0,0,33,33);
    draw_sprite(buffer.buffer,pacman,px,py);//para respetar transparencia

}
void Covid_man::atajo(int numMapa){
    if(numMapa==1){//Deinicion del atajo del mapa 1
```

```
        if(px<=-30 && py==270){
            px=870;py=270;
        }
        else if(px>=870 && py==270){
            px=-30;py=270;
        }
        else if(px<=-30 && py==540){
            px=870;py=30;
        }
        else if(px>=870 && py==30){
            px=-30;py=540;
        }
        else if(px>=870 && py==330){
            px=870;py=540;
        }
        else if(px>=870 && py==540){
            px=870;py=330;
        }
   }
   else if(numMapa==2){
        if(px<=-30 && py==270){
            px=870;py=270;
        }
        else if(px>=870 && py==270){
            px=-30;py=270;
        }
        else if(px<=-30 && py==540){
            px=870;py=30;
        }
        else if(px>=870 && py==30){
            px=-30;py=540;
        }
        else if(px>=870 && py==330){
            px=870;py=540;
            dir=0;//Para que vaya en la direccion correcta ya que el atajo entrada
//y salida estan en el mismo muro
        }
        else if(px>=870 && py==540){
            px=870;py=330;
            dir=0;//Para que vaya en la direccion correcta
        }
   }
}
int Covid_man::getPosX(){
    return px;
}
int Covid_man::getPosY(){
    return py;
}
void Covid_man::setNumMapa(int numMapa){//pARA SABER EN QUE MAPA ESTAMOS
    this->numMapa=numMapa;
}
int Covid_man::getVida(){
   return this->vidas;
}
void Covid_man::setVida(int vidas){
   this->vidas=vidas;
}
```

```
void Covid_man::ganarVida(int puntos){
    if(puntos%100==0 && puntos!=0){
        vidas++;
    }
}
```

Enfermo.h

```
#include <cstdlib>
#include "Personaje.h"
class Enfermo : public Personaje{
       private:
    BITMAP *enemigobmp=NULL;//Creara un espacio en donde estara el enemigo
    BITMAP *enemigo=NULL;//Dimensiones de un pacman;//Imagen del enemigo
    int E_dir;
    int E_x,E_y; //coordenadas    en el mapa
    int numMapa;
    public:
    Enfermo();
    Enfermo(const char*);
    ~Enfermo();
    int getPosX();
    int getPosY();
    void setNumMapa(int);
    void dibujar();
    void movimiento();
    void atajo(int);

};
```

Enfermo.cpp

```
#include "Enfermo.h"
Enfermo::Enfermo(const char* nombre){

    this->enemigobmp=load_bitmap(nombre,NULL);//Espacio en donde estara el enemigo
    this->enemigo=create_bitmap(30,30);
    this->E_dir=0;
    this->E_x=30*14;
    this->E_y=30*15; //coordenadas      en el mapa
    this->numMapa=1;
}
Enfermo::Enfermo(){
    this->enemigobmp=load_bitmap("enemigo1.bmp",NULL);//Espacio donde estará
    this->enemigo=create_bitmap(30,30);//Dimensiones de un pacman
    this->E_dir=0;
    this->E_x=30*14;
    this->E_y=30*15; //coordenadas      en el mapa
    this->numMapa=1;
}
Enfermo::~Enfermo(){

}
void Enfermo::setNumMapa(int numMapa){
   this->numMapa=numMapa;
}
int Enfermo::getPosX(){
```

```
   return E_x;
}
int Enfermo::getPosY(){
   return E_y;
}
void Enfermo::dibujar(){
       blit(enemigobmp,enemigo,0,0,0,0,30,30);
    draw_sprite(buffer.buffer,enemigo,E_x,E_y);//para respetar transparencia
}
void Enfermo::movimiento(){
       Enfermo::dibujar();

       if(numMapa==1){
        if (E_dir==0){
                if( mapa1 [E_y/30][(E_x-30)/30] != 'X')
                      E_x-=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
        }
           else if(E_dir==1){
                if( mapa1 [E_y/30][(E_x+30)/30] != 'X')
                      E_x+=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
              }

           else if(E_dir==2){
                if( mapa1 [(E_y-30)/30][E_x/30] != 'X')
                      E_y-=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
           }

           else if(E_dir==3){
                if( mapa1 [(E_y+30)/30][E_x/30] != 'X')
                      E_y+=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
           }
       }//fin numMapa==1
       else if(numMapa==2){
        if (E_dir==0){
                if( mapa2 [E_y/30][(E_x-30)/30] != 'X')
                      E_x-=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
        }
           else if(E_dir==1){
                if( mapa2 [E_y/30][(E_x+30)/30] != 'X')
                      E_x+=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
              }

           else if(E_dir==2){
                if( mapa2 [(E_y-30)/30][E_x/30] != 'X')
                      E_y-=30;
                else
```

```
                        E_dir=rand()%4; //valor al azar entre 0 y 3
          }

          else if(E_dir==3){
                if( mapa2 [(E_y+30)/30][E_x/30] != 'X')
                      E_y+=30;
                else
                      E_dir=rand()%4; //valor al azar entre 0 y 3
          }
       }//fin numMapa==2
       // movimiento para los atajos
       Enfermo::atajo(numMapa);
}
void Enfermo::atajo(int numMapa){
    if(numMapa==1){//Deinicion del atajo del mapa 1
        if(E_x<=-30 && E_y==270){
            E_x=870;E_y=270;
        }
        else if(E_x>=870 && E_y==270){
            E_x=-30;E_y=270;
        }
        else if(E_x<=-30 && E_y==540){
            E_x=870;E_y=30;
        }
        else if(E_x>=870 && E_y==30){
            E_x=-30;E_y=540;
        }
        else if(E_x>=870 && E_y==330){
            E_x=870;E_y=540;
        }
        else if(E_x>=870 && E_y==540){
            E_x=870;E_y=330;
        }
   }
   else if(numMapa==2){
        if(E_x<=-30 && E_y==270){
            E_x=870;E_y=270;
        }
        else if(E_x>=870 && E_y==270){
            E_x=-30;E_y=270;
        }
        else if(E_x<=-30 && E_y==540){
            E_x=870;E_y=30;
        }
        else if(E_x>=870 && E_y==30){
            E_x=-30;E_y=540;
        }
        else if(E_x>=870 && E_y==330){
            E_x=870;E_y=540;
            E_dir=0;//Para que vaya en la direccion correcta
        }
        else if(E_x>=870 && E_y==540){
            E_x=870;E_y=330;
            E_dir=0;//Para que vaya en la direccion correcta
        }
   }
}
```

Mapa.h

```
#include <allegro.h>
#include "Covid_man.cpp"
#include "Casa.cpp"
#include "Vacuna.cpp"
#include <iostream>
#include <cstdlib>
//buffer y el mapa tienen las mismas dimensiones
using namespace std;
const char*
CantPuntos[]={"0","1","2","3","4","5","6","7","8","9","10","11","12","13","14",
"15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","30",
"31","32","33","34","35","36","37","38","39","40","41","42","43","44","45","46",
"47","48","49","50","51","52","53","54","55","56","57","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","201",
"212","213","214","215","216","217","218","219","220"};

class Mapa{
    private:
    bool salida=false;
    bool entrar=true;
    //musica
    SAMPLE *muerte= load_wav("muerte.wav");
    MIDI *musica= load_midi("musica_fondo.mid");
    SAMPLE *recoger= load_wav("recoger.wav");
    BITMAP *fondo1=load_bitmap("FONDO1.bmp",NULL);
    BITMAP *fondo2=load_bitmap("FONDO2.bmp",NULL);
    BITMAP *fondo3=load_bitmap("FONDO3.bmp",NULL);
    BITMAP *cursor=load_bitmap("cursor.bmp",NULL);
    BITMAP *over=load_bitmap("over.bmp",NULL);
    BITMAP *win=load_bitmap("win.bmp",NULL);
    BITMAP *pausa=load_bitmap("PAUSA.bmp",NULL);

    int nivel;//nivel en el que se encuentra el mapa
    int ejeX;
    int ejeY;
    int puntos=0;
    Covid_man jugador;
    Casa* casa=new Casa();

    Pastilla* pastilla=new Pastilla();
    Vacuna* vacuna=new Vacuna();
    Enfermo enemigo1{"enemigo1.bmp"};
    Enfermo enemigo2{"enemigo2.0.bmp"};
    Enfermo enemigo3;
    public:
    Mapa(int,int,int);//(nombrefichero,nivel,ejex,ejey)
    ~Mapa();
```

```
    void Mostrar();//mOSTRAMOS TODO EL COMPORTAMIENTO DEL JUEGO EN LA PANTALLA
    void dibujar_mapa();//Creamos el mapa con todos sus componentes
    void pantalla();//Creamos el buffer enciam de la ventana
    void Proceso();//mOSTRAMOS TODO EL COMPORTAMIENTO DEL JUEGO EN LA PANTALLA
    bool Win();//Nos indica si hemos ganado un determinado nivel
    void avanzarNivel();//Para avanzar a un siguiente nivel
    const char* pantallaNivel();//dEFINIMOS EL NIVEL EN EL QUE ESTA EL MAPA
    void mostrarNivelMapa();//Mostramos el nivel del mapa en pantalla
    const char* cantidadPuntos();//Nos devolvera cuantos puntos tiene el jugador
    void imprimirPuntosPantalla();//Imprimiremos los puntos en pantalla
    void mostrarPausa();
    void cruce();
    void mostrarWin(bool);
    void mostrarOver(bool);
    void menu();

    void setNivelMapa(int);
    int getNivelMapa();
    void setEjeX(int);
    int getEjeX();
    void setEjeY(int);
    int getEjeY();
    int getPuntos();
    void setPuntos(int);
};
```

Mapa.cpp

```
#include "Mapa.h"

Mapa::Mapa(int nivel,int ejeX,int ejeY){
   this->ejeX=ejeX;
   this->ejeY=ejeY;
   //create_bitmap() crea un bitmap en la RAM
   buffer.buffer=create_bitmap(this->ejeX,this->ejeY);//(29*30,20*30) creamos un
espacio rectangular
   this->nivel=nivel;
}
void Mapa::setNivelMapa(int nivel){
   this->nivel=nivel;
}
int Mapa::getNivelMapa(){
   return this->nivel;
}
void Mapa::setPuntos(int puntos){
   this->puntos=puntos;
}
int Mapa::getPuntos(){
   return this->puntos;
}
void Mapa::setEjeX(int ejeX){
   this->ejeX=ejeX;
}
int Mapa::getEjeX(){
   return this->ejeX;
}
void Mapa::setEjeY(int ejeY){
   this->ejeY=ejeY;
```

```
}
int Mapa::getEjeY(){
   return this->ejeY;
}
void Mapa::Mostrar(){
    Mapa::Proceso();
}
void Mapa::dibujar_mapa(){//Dibujamos el mapa de acuerdo al nivel
     if(Mapa::getNivelMapa()==1){
        jugador.setNumMapa(Mapa::getNivelMapa());//Accedemos a la configuracion del
atajo del mapa1
        enemigo1.setNumMapa(Mapa::getNivelMapa());
        enemigo2.setNumMapa(Mapa::getNivelMapa());
        enemigo3.setNumMapa(Mapa::getNivelMapa());
        for(int row=0;row<MAXFILAS;row++){
          for(int col=0;col<MAXCOLS;col++){
              if(mapa1[row][col]=='X'){//Coordenada para dibujar una casa
                  casa->setTipo(row);
                  casa->definirCasa();

draw_sprite(buffer.buffer,casa->getCasa(),col*casa->getEjeX(),row*casa->getEjeY());//I
mprimimos casa sobre buffer

              }
              else if(mapa1[row][col]=='o'){//Coordenada para dibujar comida

draw_sprite(buffer.buffer,pastilla->getPastilla(),col*pastilla->getEjeX(),row*pastilla
->getEjeY());
                  if((jugador.getPosY())/30==row &&
(jugador.getPosX())/30==col){//Dividimos entre 30 para que regrese a las dimensiones
que le corresponde
                     play_sample(recoger,300,150,1000,0);
                     mapa1[row][col]=' ';
                     puntos++;
                  }
              }
              else if(mapa1[row][col]=='v'){

draw_sprite(buffer.buffer,vacuna->getVacuna(),col*vacuna->getEjeX(),row*vacuna->getEje
Y());
                  if((jugador.getPosY())/30==row &&
(jugador.getPosX())/30==col){//Dividimos entre 30 para que regrese a las dimensiones
que le corresponde
                     play_sample(recoger,300,150,1000,0);
                     mapa1[row][col]=' ';
                     puntos=puntos+2;
                  }
              }
          }
        }
     }//fin nivel1
     else if(Mapa::getNivelMapa()==2){
        jugador.setNumMapa(Mapa::getNivelMapa());//Accedemos a la configuracion del
atajo del mapa2
        enemigo1.setNumMapa(Mapa::getNivelMapa());
        enemigo2.setNumMapa(Mapa::getNivelMapa());
        enemigo3.setNumMapa(Mapa::getNivelMapa());
        for(int row=0;row<MAXFILAS;row++){
```

```
          for(int col=0;col<MAXCOLS;col++){
              if(mapa2[row][col]=='X'){
                  casa->setTipo(row);
                  casa->definirCasa();

draw_sprite(buffer.buffer,casa->getCasa(),col*casa->getEjeX(),row*casa->getEjeY());
              }
              else if(mapa2[row][col]=='o'){

draw_sprite(buffer.buffer,pastilla->getPastilla(),col*pastilla->getEjeX(),row*pastilla
->getEjeY());
                  if((jugador.getPosY())/30==row &&
(jugador.getPosX())/30==col){//Dividimos entre 30 para que regrese a las dimensiones
que le corresponde
                     play_sample(recoger,300,150,1000,0);
                     mapa2[row][col]=' ';
                     puntos++;
                  }
              }
              else if(mapa2[row][col]=='v'){

draw_sprite(buffer.buffer,vacuna->getVacuna(),col*vacuna->getEjeX(),row*vacuna->getEje
Y());
                  if((jugador.getPosY())/30==row &&
(jugador.getPosX())/30==col){//Dividimos entre 30 para que regrese a las dimensiones
que le corresponde
                     play_sample(recoger,300,150,1000,0);
                     mapa2[row][col]=' ';
                     puntos=puntos+2;
                  }
              }


          }
        }
     }//fin nivel2
   }
void Mapa::pantalla(){//blit() lo copia a la pantalla
    blit(buffer.buffer,screen,0,0,0,0,buffer.ejeX,buffer.ejeY);//imprimimos el buffer
sobre la pantalla
    //buffer origen y screen destino
    //Copia un área rectangular del mapa de bits de origen en el mapa de bits de
destino.
}//para mi
void Mapa::Proceso(){
    pantalla();
    menu();//pantalla de inicio del usuario
    play_midi(musica,0);
    while(!key[KEY_ESC] && entrar==true){

         jugador.movimiento();
         jugador.ganarVida(puntos);
         Mapa::dibujar_mapa();//creamos el buffer

         Mapa::mostrarNivelMapa();//Colocamos aqui para que no aprezca de forma
parpadeante en pantalla
         Mapa::imprimirPuntosPantalla();
         Mapa::mostrarPausa();
```

```
        textout_centre_ex(buffer.buffer, font,"VIDAS:", 210, 630, 0xFFFFFF, 0);
        textout_centre_ex(buffer.buffer,
font,jugador.imprimirVida(jugador.getVida()), 190, 640, 0xFFFFFF, 0);

        //Zona de declaracion de jugadores
        enemigo1.movimiento();
        enemigo2.movimiento();
        enemigo3.movimiento();
        jugador.dibujar();

        Mapa::cruce();//Cruce entre jugador y enemigos

        pantalla();
        rest(70);//delay 70 ms
        jugador.morderPacman();
        pantalla();

        rest(90);
        Mapa::avanzarNivel();

        Mapa::mostrarOver(entrar);
        Mapa::mostrarWin(entrar);

        if(key[KEY_P]){
           clear(buffer.buffer);
           //textout_centre_ex(buffer.buffer, font,"WIN ", 390, 390, 0xFFFFFF, 0);
           blit(pausa,buffer.buffer,0,0,0,0,1280,930);
           pantalla();
           rest(90);
           clear_keybuf();
           while(true){
               if(key[KEY_P]){
                   break;
               }
           }
        }
    }
}
void Mapa::menu(){
    while(!salida){
        if(mouse_x>380 && mouse_x<500 && mouse_y>290 && mouse_y<345){//comenzar
            blit(fondo2,buffer.buffer,0,0,0,0,880,700);
            if(mouse_b& 1){//1 es click derrecho mouse_b es click
                salida=true;
            }
        }
        else if(mouse_x>380 && mouse_x<500 && mouse_y>400 && mouse_y<460){
            blit(fondo3,buffer.buffer,0,0,0,0,880,700);
            if(mouse_b& 1){//1 es click derrecho mouse_b es click
                salida=true;
                entrar=false;
            }
        }
        else{
            blit(fondo1,buffer.buffer,0,0,0,0,880,700);
        }
        masked_blit(cursor,buffer.buffer,0,0,mouse_x,mouse_y,13,22);
        blit(buffer.buffer,screen,0,0,0,0,880,700);
```

```
        clear(buffer.buffer);
    }
}
void Mapa::mostrarWin(bool entrar){
    if(Mapa::getNivelMapa()>2 && jugador.getVida()>0){
            clear(buffer.buffer);
            //textout_centre_ex(buffer.buffer, font,"WIN ", 390, 390, 0xFFFFFF, 0);
            blit(win,buffer.buffer,0,0,0,0,1280,930);
            remove_sound();
            pantalla();
            rest(10000);//Esperamos 10 segundos
            entrar=false;
    }
}
void Mapa::mostrarOver(bool entrar){
    if(jugador.getVida()==0){//Si perdemos el juego
            clear(buffer.buffer);
            //textout_centre_ex(buffer.buffer, font,"GAME OVER ", 390, 390, 0xFFFFFF,
0);
            blit(over,buffer.buffer,0,0,0,0,1280,930);
            remove_sound();
            pantalla();
            rest(10000);//Esperamos 10 segundos
            entrar=false;
      }
}
void Mapa::cruce(){
    if(jugador.getPosX()==enemigo1.getPosX() &&
jugador.getPosY()==enemigo1.getPosY()){
           //play_sample(jugador.getMusicMuerte(),300,150,1000,0);
           rest(70);
           jugador.setVida(jugador.getVida()-1);
           for(int i=0;i<6;i++){
              jugador.inicioMuerte();
              Mapa::dibujar_mapa();
              jugador.finalMuerte(i);

             pantalla();
             rest(70);
           }
           //play_midi(musica,1);
       }
    if(jugador.getPosX()==enemigo2.getPosX() &&
jugador.getPosY()==enemigo2.getPosY()){
            play_sample(jugador.getMusicMuerte(),300,150,1000,0);
            rest(70);
            jugador.setVida(jugador.getVida()-1);
           for(int i=0;i<6;i++){
              jugador.inicioMuerte();
              Mapa::dibujar_mapa();
              jugador.finalMuerte(i);

             pantalla();
             rest(70);
           }
           //play_midi(musica,1);
       }
    if(jugador.getPosX()==enemigo3.getPosX() &&
```

```
jugador.getPosY()==enemigo3.getPosY()){
            play_sample(jugador.getMusicMuerte(),300,150,1000,0);
            rest(70);
            jugador.setVida(jugador.getVida()-1);
           for(int i=0;i<6;i++){
              jugador.inicioMuerte();
              Mapa::dibujar_mapa();
              jugador.finalMuerte(i);
             pantalla();
             rest(70);
           }
       }
}
bool Mapa::Win(){//Para saber si hemos ganado
    int row,col;
    if(Mapa::getNivelMapa()==1){
      for(row=0;row<MAXFILAS;row++){
         for(col=0;col<MAXCOLS;col++){
            if(mapa1[row][col]=='o')
                return false;
         }
      }
    }
    else if(Mapa::getNivelMapa()==2){
      for(row=0;row<MAXFILAS;row++){
         for(col=0;col<MAXCOLS;col++){
            if(mapa2[row][col]=='o')
                return false;
         }
      }
    }

   return true;

}
void Mapa::avanzarNivel(){
   if(Mapa::Win()){
      Mapa::setNivelMapa(++nivel);//Para pasar al siguiente nivel
   }

}
const char* Mapa::pantallaNivel(){
   const char pantNivel[]="0";
   const char *pantNivelPtr=pantNivel;
   if(Mapa::getNivelMapa()==1){
       pantNivelPtr="1";
   }
   else if(Mapa::getNivelMapa()==2){
       pantNivelPtr="2";
   }
   else if(Mapa::getNivelMapa()==3){
       pantNivelPtr="3";
   }
   return pantNivelPtr;
}
void Mapa::mostrarNivelMapa(){
   textout_centre_ex(buffer.buffer, font,"NIVEL:", 30, 630, 0xFFFFFF, 0);
   textout_centre_ex(buffer.buffer, font,Mapa::pantallaNivel(), 10, 640, 0xFFFFFF, 0);
```

```
}
void Mapa::imprimirPuntosPantalla(){
   textout_centre_ex(buffer.buffer, font,"PUNTAJE:", 120, 630, 0xFFFFFF, 0);
   textout_centre_ex(buffer.buffer, font,Mapa::cantidadPuntos(), 90, 640, 0xFFFFFF,
0);
}
void Mapa::mostrarPausa(){
      textout_centre_ex(buffer.buffer, font,"PAUSAR CON LA LETRA 'P' ",350, 630,
0xFFFFFF, 0);
}
const char* Mapa::cantidadPuntos(){
    return CantPuntos[Mapa::getPuntos()];
}
Mapa::~Mapa(){
      delete pastilla;
      delete  vacuna;
}
```

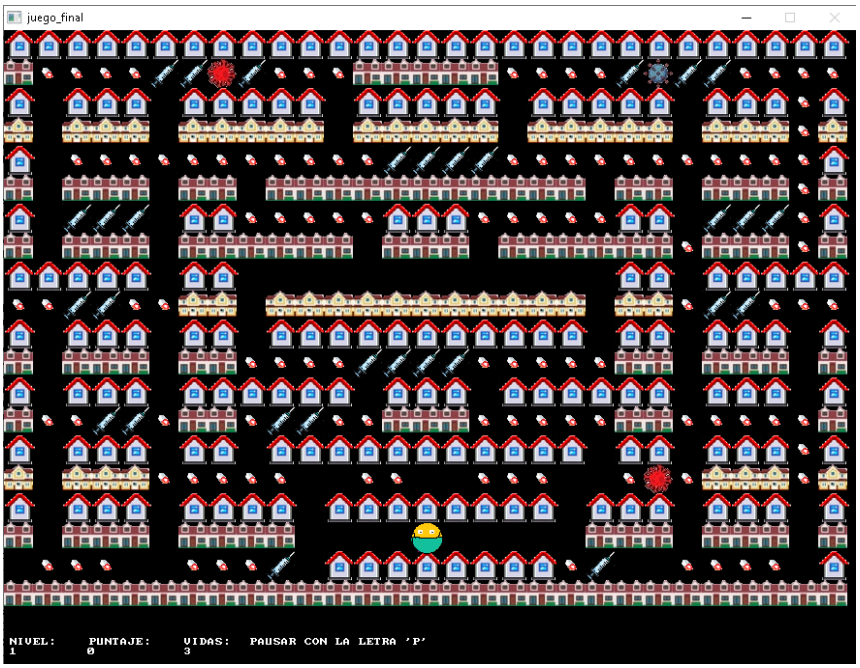
main.cpp

```
#include <allegro.h>
#include "Mapa.cpp"

int main()
{
    allegro_init();//inicia la librería Allegro
    install_keyboard();//nos permite utilizar las teclas
    install_mouse();//parametros del mouse
    set_color_depth(32);

    //Inicializacion de la instancia Nivel 1
    int nivel=1;//Elegimos el nivel 1 del juego
    int ejeX=880;//Dimensiones de la ventana
    int ejeY=700;

    set_gfx_mode(GFX_AUTODETECT_WINDOWED,ejeX,ejeY,0,0);

    //codigo musica en allegro
       if(install_sound(DIGI_AUTODETECT, MIDI_AUTODETECT, NULL) != 0){
        allegro_message("Error: inicializando sistema de sonido\n%s\n",
allegro_error);
        return 1;
    }
    //control para el sonido, izq y der
       set_volume(200, 200);

    Mapa noob(nivel,ejeX,ejeY);
    noob.setNivelMapa(2);
    noob.Mostrar();
    return 0;
}
END_OF_MAIN();
```

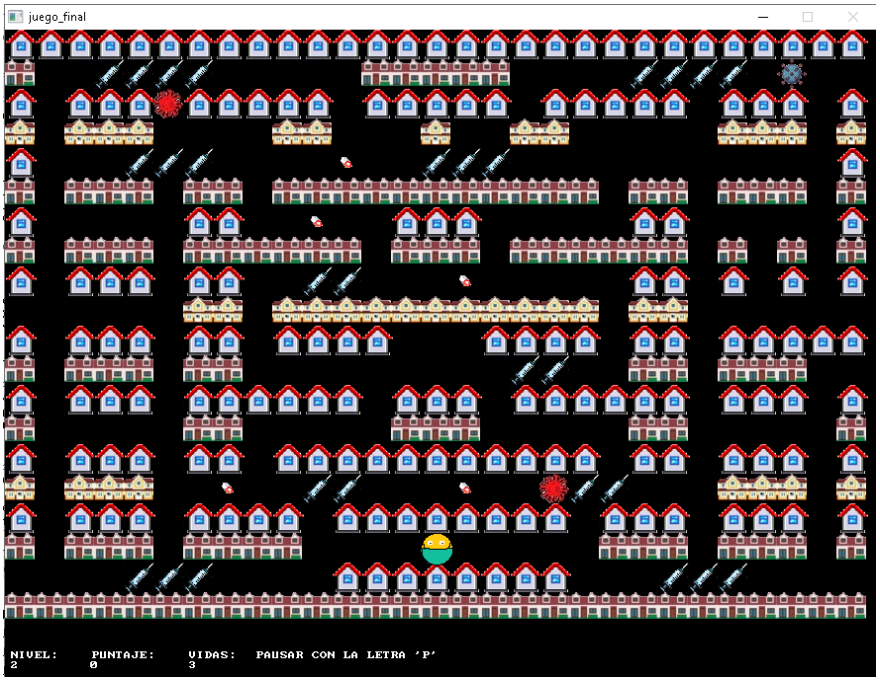
5. Interfaces

5.1. Menú principal del juego

5.2. Mapa del Primer nivel



5.3. Mapa del Segundo nivel



6. Conclusión

La totalidad del desarrollo del código del videojuego está basado en los temas abordados durante todo el semestre a excepción del uso de una interfaz gráfica mediante la librería Allegro ,se usó como principal herramienta el paradigma de programación orientado a objetos para la creación de cada elemento usado en el proyecto a través de la abstracción de estos mismos.

A través de la librería allegro pudimos aprender el manejo de una interfaz gráfica, como lo sería el manejo de buffers para la colocación del objeto en un espacio , la creación de mapas mediante arreglos , la carga de sprites para que pueda dar una apariencia a los objetos mencionados , además del manejo de sonidos para ingresarlos en situaciones específicas .

7. Bibliografía

- Curso de Programacion de videojuegos con C++ y Allegro (2001).Recuperado de : **http://wiki.joanillo.org/images/5/5e/Curso_de_Programacion_videojuegos_C%2B%2B_Allegro3.pdf**
- Una libreria para programar videojuegos. Recuperado de: **http://www.ganimides.ucm.cl/haraya/doc/Introduc_Allegro.pdf**

8. Anexos

Link del repositorio en GitHub
https://github.com/Yanira1612/PROYECT-COVID-MAN-2.0