Allegro

Knox Game Design

December 2020

Levi D. Smith

Overview

- Atari Low-Level Game Routines
- DOS game developers
- C, C++ developers
- Windows, Linux, Mac, iPhone, Android builds

KNOX GAME DESIGN

History

- Allegro 2,3,4
  - Late 1990s
  - Able to make DOS executables
  - DJGPP
    - free DOS compiler (C, C++)
    - DJ Delorie
    - http://www.delorie.com/djgpp/history.html
  - https://liballeg.org/readme.html
- Allegro 5
  - Latest version
  - Allegro 2,3,4 code not compatible with Allegro 5 (new API)

Installing

- Download
  - https://liballeg.org/download.html
- Windows
  - https://github.com/liballeg/allegro_wiki/wiki/Quickstart
    - Example code and compile commands
  - DJGPP
  - MSYS2
  - Visual Studio
    - IntelliSense/Code completion

KNOX GAME DESIGN

DJGPP

- Allegro 2/4
- ZIP Picker - http://www.delorie.com/djgpp/zip-picker.cgi
- May need CWSDPMI from https://dosgames.com/utilities.php put in djgpp/bin
- gcc tetris.c -otetris -lalleg -g


```
@echo off
set PATH=c:\djgpp\bin;%PATH%
set DJGPP=c:\djgpp\djgpp.env
```

www.delorie.com/djgpp/ search

djgpp

DJGPP is a complete 32-bit C/C++ development system for Intel 80386 (and higher) PCs running DOS. It includes ports of many GNU development utilities. The development tools require a 80386 or newer computer to run, as do the programs they produce. In most cases, the programs it produces can be sold commercially without license or royalties.

www.delorie.com/djgpp/zip-picker.cgi search

DJGPP Zip File Picker Results

FTP Site: ftp://ftp.delorie.com/pub/djgpp/current/

Note: Don't install djgpp in `/dev` as djgpp treats that directory specially.

Note: If your FTP site doesn't have the selected files, please choose another one, like ftp.delorie.com

Note: Do not install DJGPP in a directory that requires long file name support (i.e. no spaces, 8.3 characters)

Note: The FAQ has information on adjusting the registry so that one installation works with both long and short filenames (Win 95/98/NT and DOS)

Note: To view online documentation and help, run the "`info`" program.

Note: RHIDE has an online help viewer also.

Note: Start RHIDE in the directory with your sources, so that it puts the .exe there also.

Note: In RHIDE, press F1 twice to get help on help

Note: Sample compile command: `gcc hello.c -o hello.exe`

Note: You need to build Allegro before you can use it. Use "make"

Read the file `README.1ST` before you do anything else with DJGPP! It has important installation and usage instructions.

unzip32.exe	to unzip the zip files	95 kb
v2/copying.dj	DJGPP Copyright info	3 kb
v2/djdev205.zip	DJGPP Basic Development Kit	2.4 mb
v2/faq230b.zip	Frequently Asked Questions	664 kb
v2/readme.1st	Installation instructions	23 kb
v2apps/rhid15ab.zip	RHIDE	6.0 mb
v2gnu/bnu234b.zip	Basic assembler, linker	5.9 mb
v2gnu/gcc930b.zip	Basic GCC compiler	34.1 mb
v2gnu/gdb801b.zip	GNU debugger	4.4 mb
v2gnu/mak43b.zip	Make (processes makefiles)	420 kb
v2tk/allegro/all422ar2.zip	Allegro game library	2.4 mb
v2tk/allegro/all422br2.zip	Allegro game library	16.2 mb

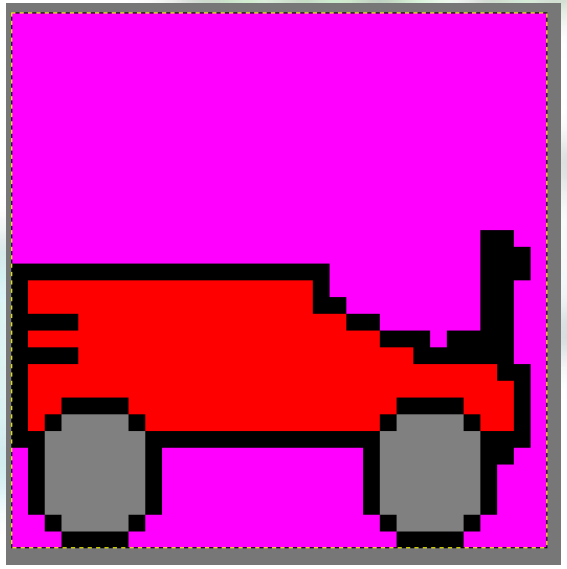
Total bytes to download: 76,173,369

Example DOS Allegro Game

KNOX GAME DESIGN

Graphics (Allegro 2,3,4)

- Used PCX image format by default
  load_pcx("<filename>", my_palette);
- Magenta (aka hot pink) for transparency color



KNOX GAME DESIGN

MSYS2

- https://www.msys2.org/
- https://www.allegro.cc/forums/thread/616828
- Download allegro-x86_64-w64-mingw32-gcc-9.2.0-posix-seh-dynamic-5.2.6.0.zip and copy files to appropriate msys2 folder
- Or run # pacman -S mingw-w64-x86_64-allegro
- Note - **msys2 is not MinGW**!

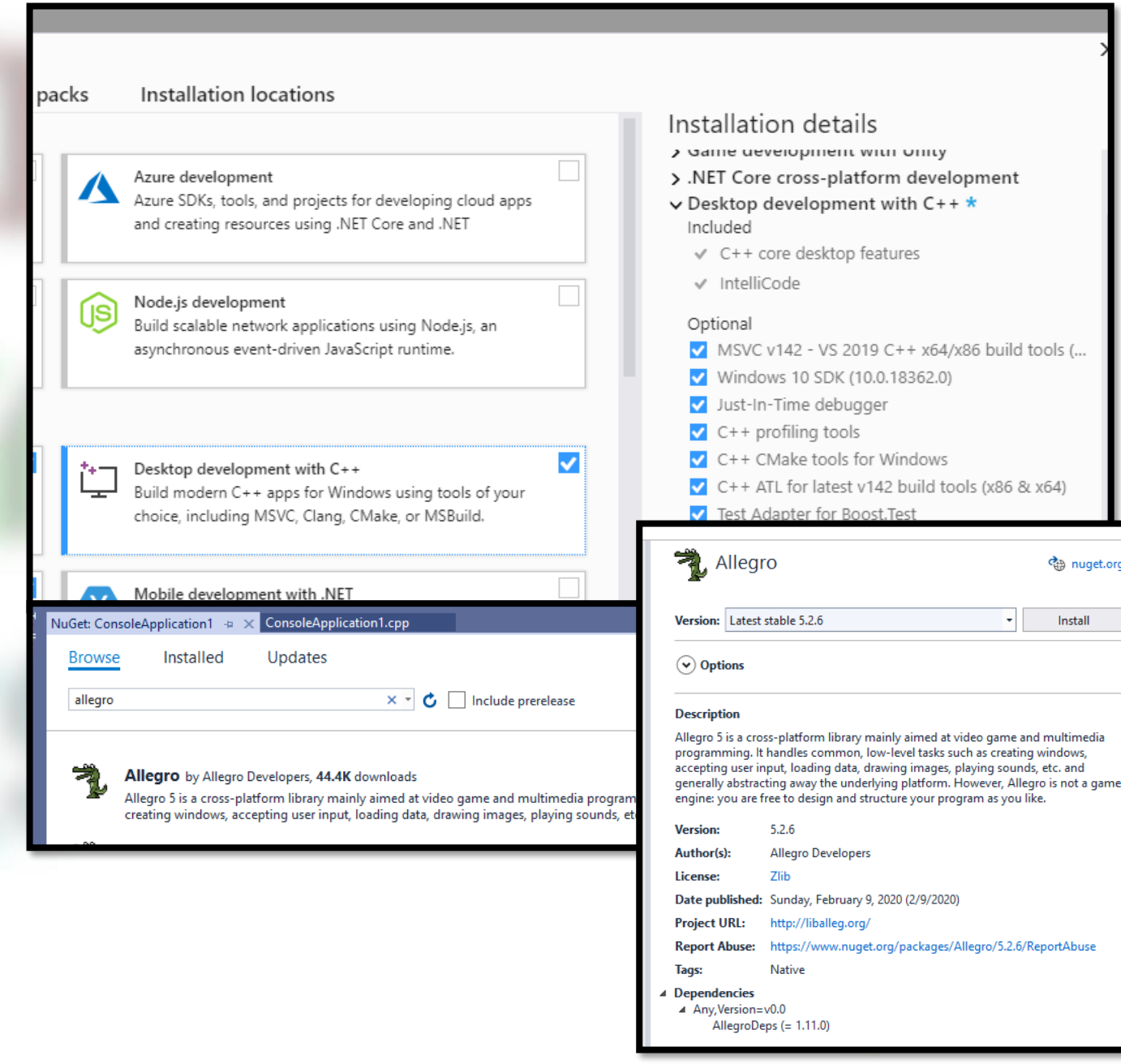
```
gatec@darknut MSYS ~
$ pacman -S base-devel gcc vim cmake
```

```
gatec@darknut MSYS ~
$ pacman -S mingw-w64-x86_64-allegro
```

KNOX GAME DESIGN

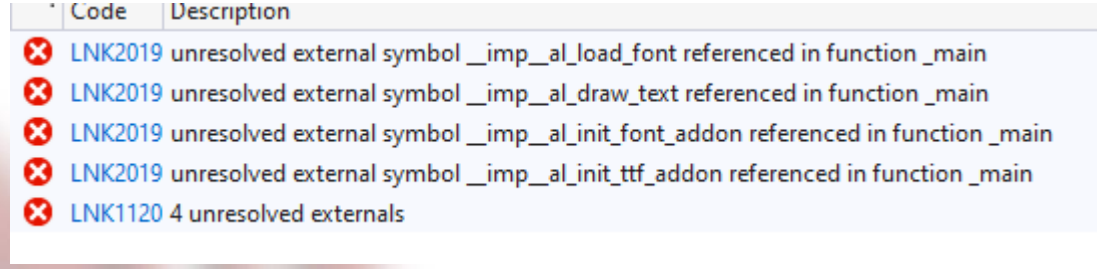
Visual Studio

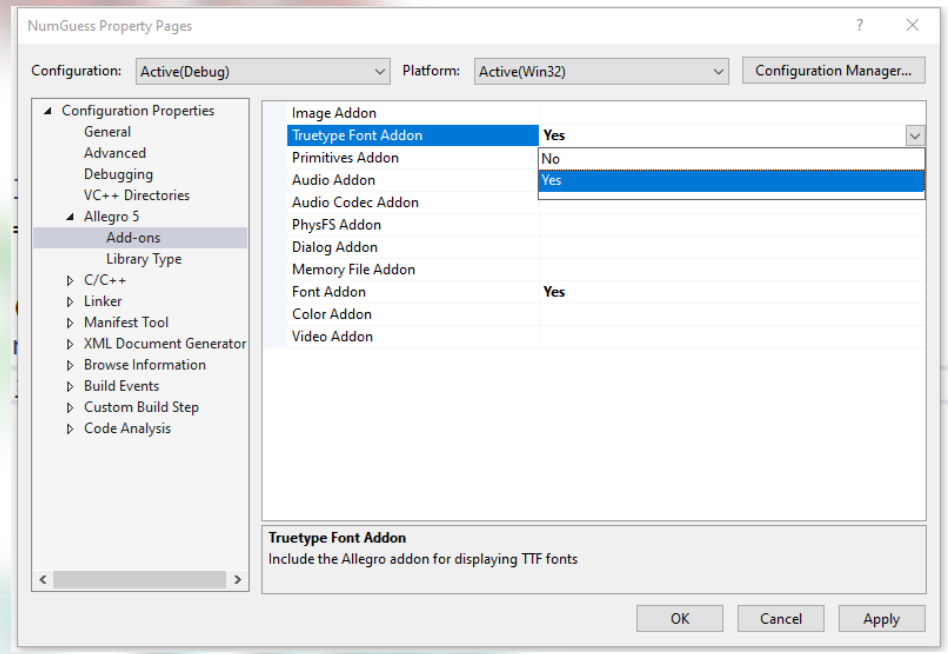
- In Visual Studio Installer (separate program), add support for *Desktop development with C++* (not installed by default)
- New C++ console application
- References > Manage NuGet Packages > Allegro > Install
- https://www.nuget.org/packages/Allegro/

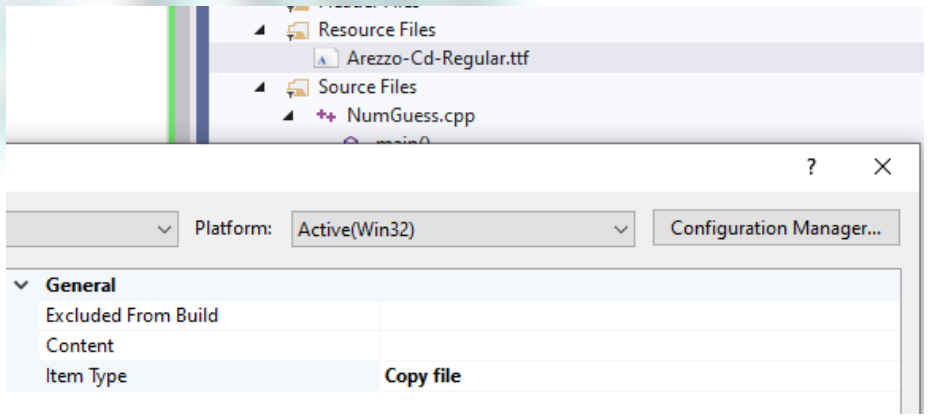


Visual Studio

- Enable Font support under project properties
  - Otherwise an error on al_create_builtin_font will occur
  - Drag font TTF file to *Resource Files* and set *Item Type* to **Copy file**
- The example Allegro program should run
  - Replace code in **ConsoleApplication1.cpp** with the example code







KNOX GAME DESIGN

Hello World

```
1   //2020 Levi D. Smith
2   #include <allegro5/allegro.h>
3   #include <allegro5/allegro_ttf.h>
4   #include <allegro5/allegro_font.h>
5
6   int main() {
7       al_init();
8       al_init_font_addon();
9       al_init_ttf_addon();
10
11      ALLEGRO_DISPLAY *display = al_create_display(800, 600);
12      ALLEGRO_FONT *myfont = al_load_font("..\\Debug\\Arezzo-Cd-Regular.ttf", 128, 0);
13      al_clear_to_color(al_map_rgb(255, 255, 0));
14      al_draw_text(myfont, al_map_rgb(0, 0, 0), 0, 0, 0, "Hello Allegro");
15
16      al_flip_display();
17      al_rest(5);  //keep the window from immediately closing
18
19      //std::cout << "Hello World!\n";
20  }
```

Tip: Use *al_get_current_directory()* to determine the current working directory (for font file)

KNOX GAME DESIGN

Keyboard Input

- Option 1 - Polling / State based
  - Must track previous state for key down/key up events

```
1    al_install_keyboard();
2    ALLEGRO_KEYBOARD_STATE kb_state;
3    al_get_keyboard_state(&kb_state);
4    if (al_key_down(&kb_state, ALLEGRO_KEY_0) {
5      cout << "0 key down" << endl;
6
7    }
```

- Option 2 - Event based
  - Get keyboard events when key is pressed or released
- Use ALLEGRO_KEY_<key> for keycode comparison
- **NOTE:** Allegro keycode values do not follow the standard ASCII numbering scheme
  - 'a' = 1
  - '0' = 27
  - <space> = 75

```
1  ALLEGRO_EVENT_QUEUE *eventqueue = NULL;
2  ALLEGRO_EVENT event;
3
4  eventqueue = al_create_event_queue();
5  al_register_event_source(eventqueue, al_get_keyboard_event_source());
6  al_wait_for_event(eventqueue, &event);
7  if (event.type == ALLEGRO_EVENT_KEY_DOWN) {
8  if (event.keyboard.keycode == ALLEGRO_KEY_0) {
9    cout << "Pressed 0" << endl;
10 }
```

GAME DESIGN

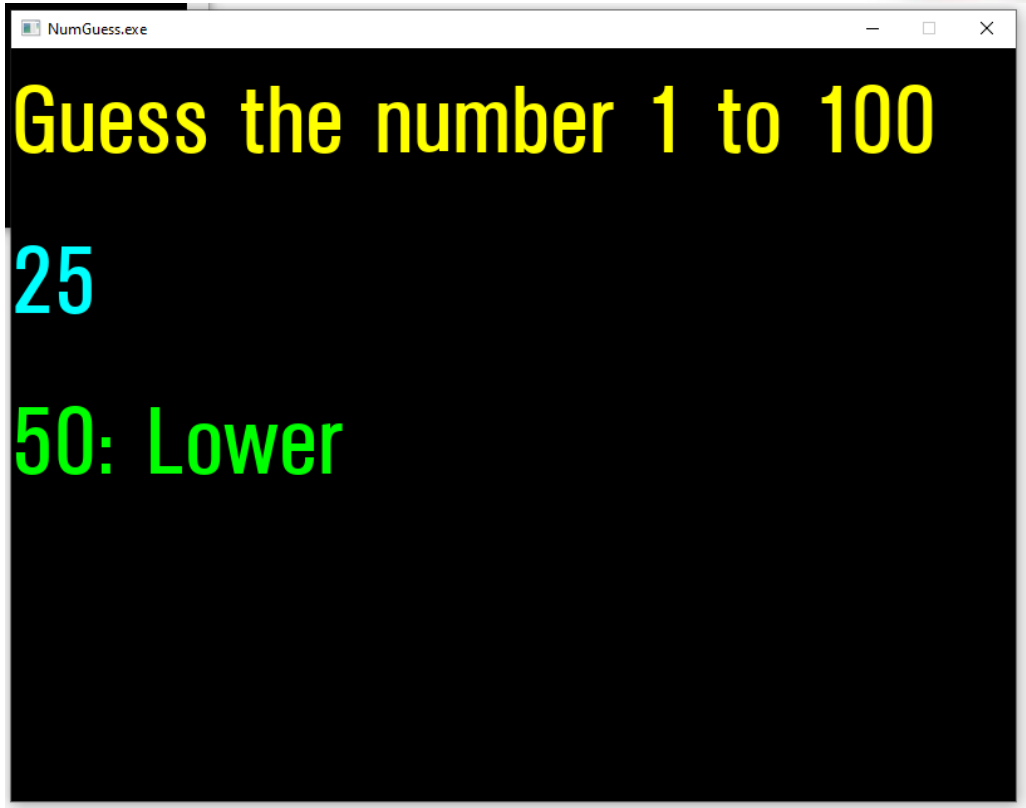
Random numbers

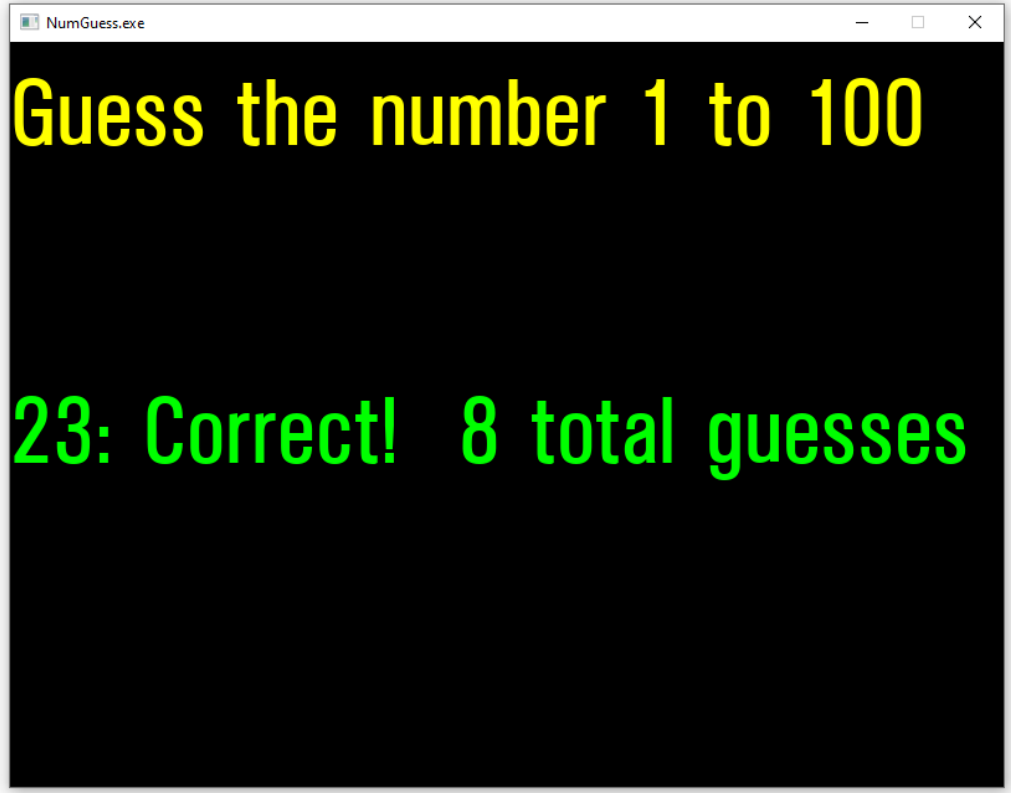
- Standard C *rand()* function works
  - #include <cstdlib>
  - rand() % 100 + 1
- C++ random library
  - #include <random>
  - random_device rd;
  - uniform_int_distribution<int> dist(1, 100);
  - dist(rd);

Number Guessing Game

```cpp
//2020 Levi D. Smith
using namespace std;
#include <iostream>
#include <string>
#include <random>
#include <allegro5/allegro.h>
#include <allegro5/allegro_ttf.h>
#include <allegro5/allegro_font.h>

int main() {
    al_init();
    al_init_font_addon();
    al_init_ttf_addon();
    al_install_keyboard();

    ALLEGRO_DISPLAY *display = al_create_display(800, 600);
    ALLEGRO_KEYBOARD_STATE kb_state;
    ALLEGRO_EVENT_QUEUE *eventqueue = NULL;
    ALLEGRO_EVENT event;

    ALLEGRO_FONT *myfont = al_load_font("..\\Debug\\Arezzo-Cd-Regular.ttf", 80, 0);
    eventqueue = al_create_event_queue();
    al_register_event_source(eventqueue, al_get_keyboard_event_source());

    random_device rd;
    uniform_int_distribution<int> dist(1, 100);

    int iSecretNumber = dist(rd);
    int iNumGuess = -1;
    int iGuessCount = 0;

    string strInput = "";
    string strResult = "";

    while (true) {
        string strMessage = "Guess the number 1 to 100";

        al_get_keyboard_state(&kb_state);
        if (al_key_down(&kb_state, ALLEGRO_KEY_ESCAPE)) {
            break;
        }

        al_clear_to_color(al_map_rgb(0, 0, 0));
        al_draw_text(myfont, al_map_rgb(255, 255, 0), 0, 0, 0, strMessage.c_str());
        al_draw_text(myfont, al_map_rgb(0, 255, 255), 0, 128, 0, strInput.c_str());
        al_draw_text(myfont, al_map_rgb(0, 255, 0), 0, 256, 0, strResult.c_str());
        al_flip_display();

        al_wait_for_event(eventqueue, &event);
        if (event.type == ALLEGRO_EVENT_KEY_DOWN) {
            int iKey = event.keyboard.keycode;
            if (iKey >= ALLEGRO_KEY_0 && iKey <= ALLEGRO_KEY_9) {
                if (strInput.size() < 3) {
                    strInput.append(to_string(iKey - ALLEGRO_KEY_0));
                }
            }

            if (iKey == ALLEGRO_KEY_BACKSPACE) {
                if (strInput.size() > 0) {
                    strInput = strInput.substr(0, strInput.size() - 1);

                }
            }

            if (iKey == ALLEGRO_KEY_ENTER) {
                try {
                    iNumGuess = stoi(strInput);
                } catch (invalid_argument &ia) {
                    cout << "invalid number" << endl;
                    continue; //skips over the rest of the loop
                }
                strResult = to_string(iNumGuess) + ": ";

                iGuessCount++;

                if (iNumGuess < iSecretNumber) {
                    strResult.append("Higher");
                } else if (iNumGuess > iSecretNumber) {
                    strResult.append("Lower");
                } else if (iNumGuess == iSecretNumber) {
                    strResult.append("Correct!  " + to_string(iGuessCount) + " total guesses");
                }

                strInput = "";
            }
        }
    }
}
```

Number Guessing Game





KNOX GAME DESIGN

C++

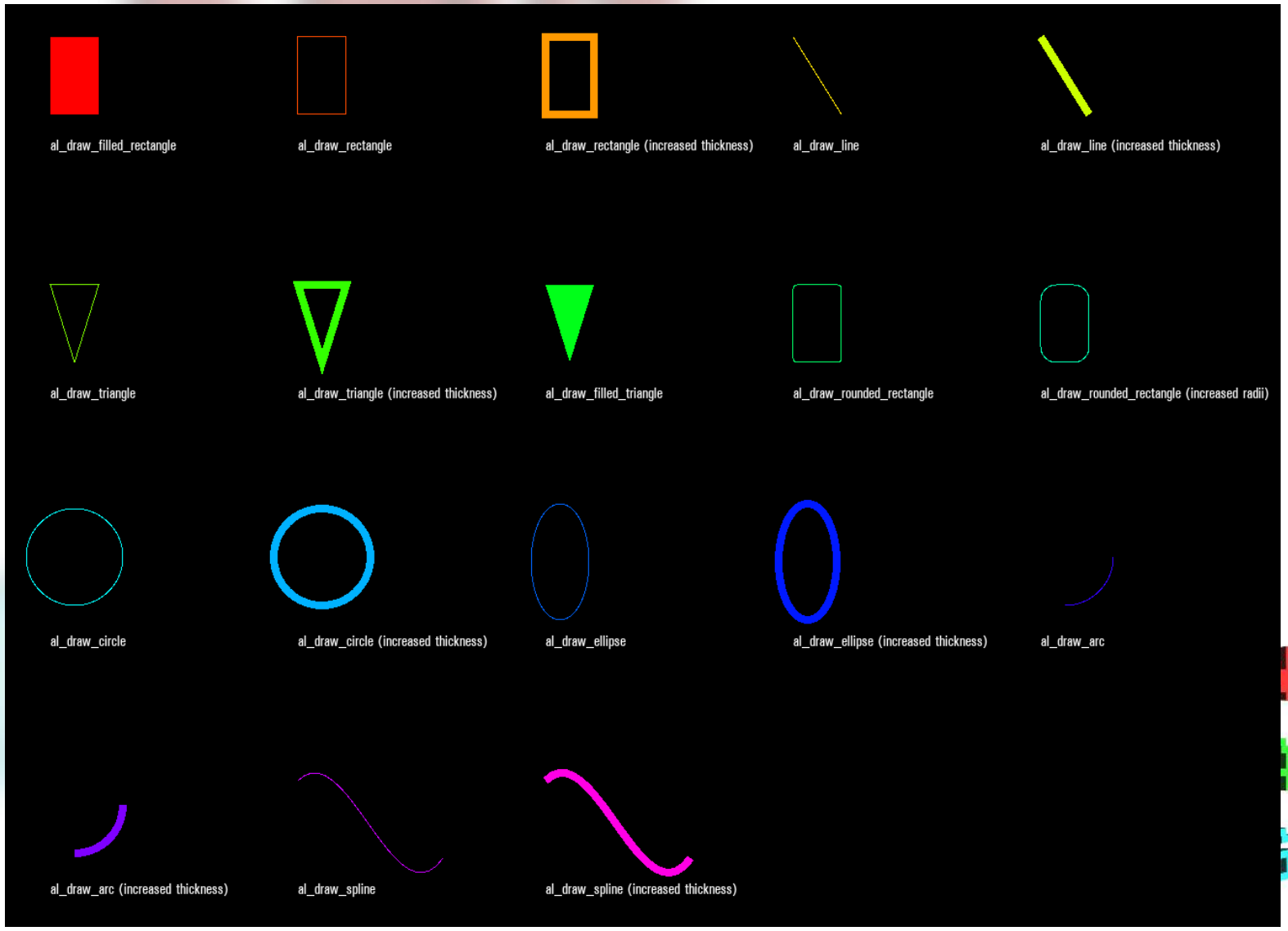
- Pros over C
  - objects, classes, polymorphism
  - No typedef structs
  - String class
  - No memory management (malloc, free)
    - But you still have to delete objects
  - Vectors for lists, dynamically sized arrays
  - bool (boolean) data type; *true* and *false* values
- Cons to Java/C#
  - Header files
    - Separate file from code
    - Function declarations; keeping compiler happy
  - Having to #include other class header files
  - Still requires old style C main entry point
  - Uses C style pointer notation (Class *pointer, object->method)

Tip: Must use keyword "class" when referencing one class from another class header file

Tip: Use *using namespace std;* when using vector to avoid typing *std::vector*

Drawing Primitives

- Note - primitives (such as al_draw_rectangle) require the Allegro primitives library to be included
- Also be sure to call *ai_init_primitives_addon()*, or you will get access violation errors



Sound Effects

- Enable *AudioAddon* and *AudioCodecAddon* in project properties
- Include *allegro_audio.h* and *allegro_acodec.h*
- Initialize with *al_install_audio()* and *al_init_acodec_addon()*
- Use *ALLEGRO_SAMPLE ** for sound data
- Drag audio files into Visual Studio under *Resources*
- Must call *al_reserve_samples(int)* before loading samples
- Load samples with *al_load_sample(<filename>)*
- *al_play_sample* to play sound
- Tip - make sure to have all sound loading code before *al_create_display*

```
al_play_sample(gamemanager->soundShipDead, 1, ALLEGRO_AUDIO_PAN_NONE, 1, ALLEGRO_PLAYMODE_ONCE, NULL);
```

Music

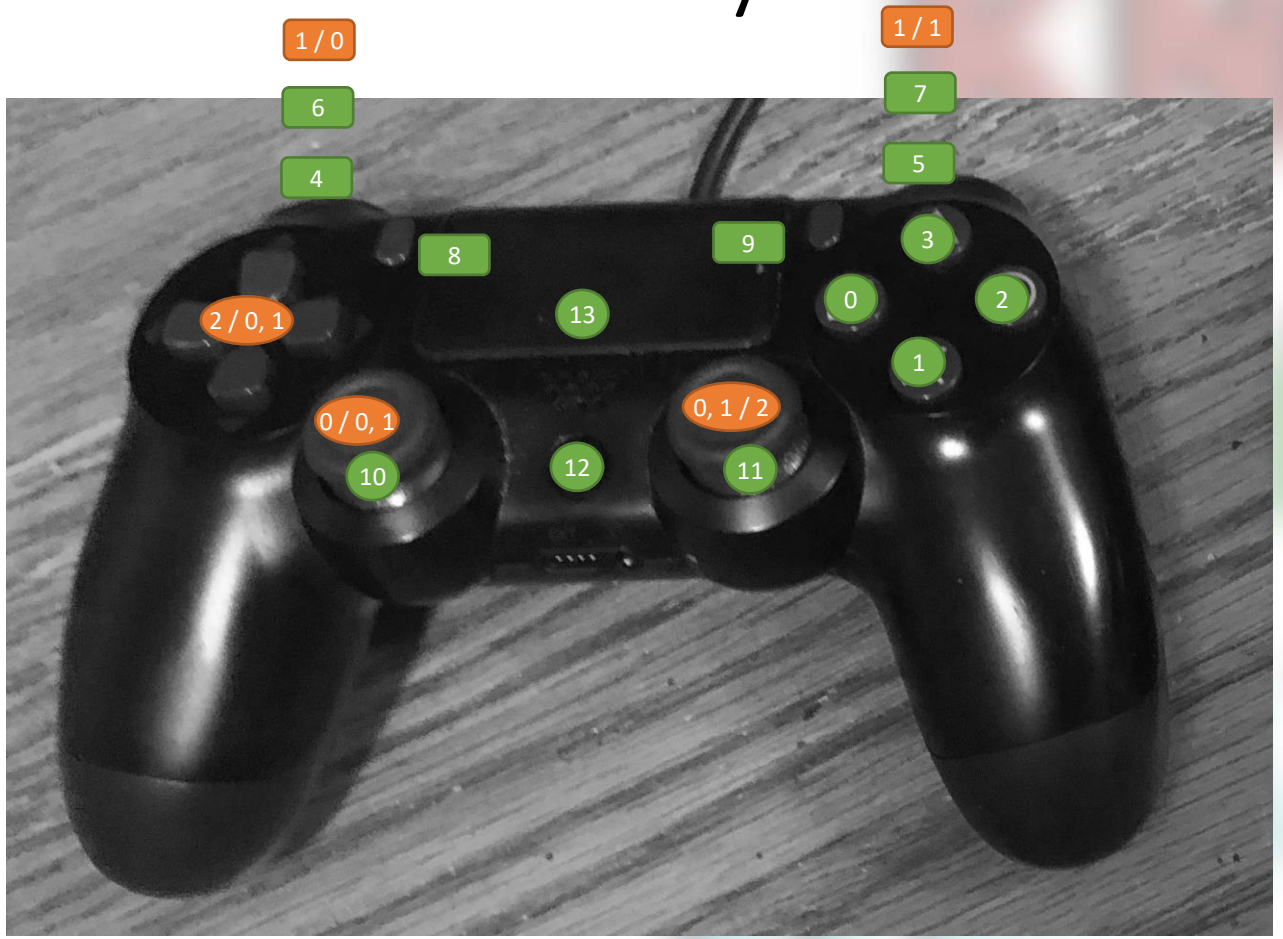
- Looping a song

```
ALLEGRO_SAMPLE *song = al_load_sample("..\\theme.wav");
ALLEGRO_SAMPLE_INSTANCE *songInstance = al_create_sample_instance(song);
al_set_sample_instance_playmode(songInstance, ALLEGRO_PLAYMODE_LOOP);
al_attach_sample_instance_to_mixer(songInstance, al_get_default_mixer());
al_play_sample_instance(songInstance);
```

Gamepad / Joystick controls

- No additional libraries needed
- Start with *al_install_joystick*
  - Note - Allegro seems buggy about order of install calls. Be sure to call joystick install before audio install
- Event based
  - *al_register_event_source(eventqueue, al_get_joystick_event_source())*
  - Can have multiple joysticks
    - *event.joystick.id == al_get_joystick(<int>)*
  - Can have multiple sticks on a joystick
    - left = 0, right = 1, D-pad = 2 - *event.joystick.stick == <int>*
  - Determine axis with *event.joystick.axis* (x = 0, y = 1)
  - Axis value *event.joystick.pos*
  - Note - stick will probably generate multiple events per update, so you must handle all events, until *al_get_next_event* is NULL
  - Buttons - *event.joystick.button == <int>*
- Polling based
  - *al_get_joystick_state(<ALLEGRO_JOYSTICK *>, <ALLEGRO_JOYSTICK_STATE *>)*
  - Seems more stable than event based, especially with two active axis
  - Downside is handling button up / button down
- Can use mix of event based and polling based
- Note - Should check if joystick exists before joystick handling code
- Note - Be careful with mixing joystick polling with keyboard move events.

Controller Layout



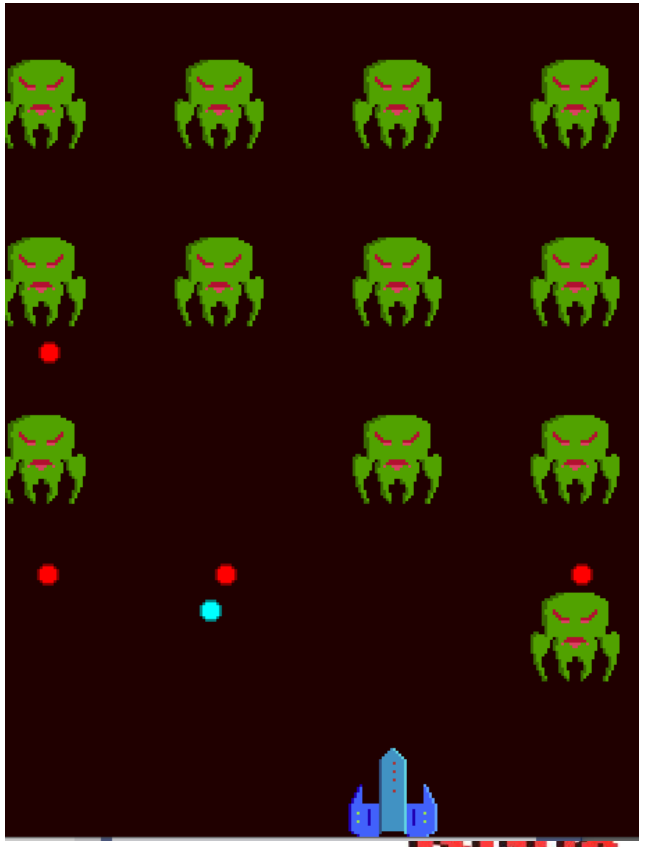
= button

= stick / axis

GAME DESIGN

Displaying Sprites

- Enable **Image Addon** in properties, Include *<allegro5/allegro_image.h>*, and call *al_init_image_addon();*
- Create <ALLEGRO_BITMAP *> to hold a sprite
- Load with *al_load_bitmap(<filename>)*
  - Works with **.BMP**, **.PNG**, **.PCX** (and others?)
  - Preserves PNG transparency
- Copy files to *Resource Files* folder in Visual Studio, set *Copy File* in properties
- Replace rectangle draw code with *al_draw_bitmap(<sprite>, x, y, 0);*
- Add countdown to switch between animation frames
  - Probably needs to be a separate animation class



Old School Transparency

- Converts magenta pixels to transparent
- Not optimized, probably slow. Need to lock bitmap?

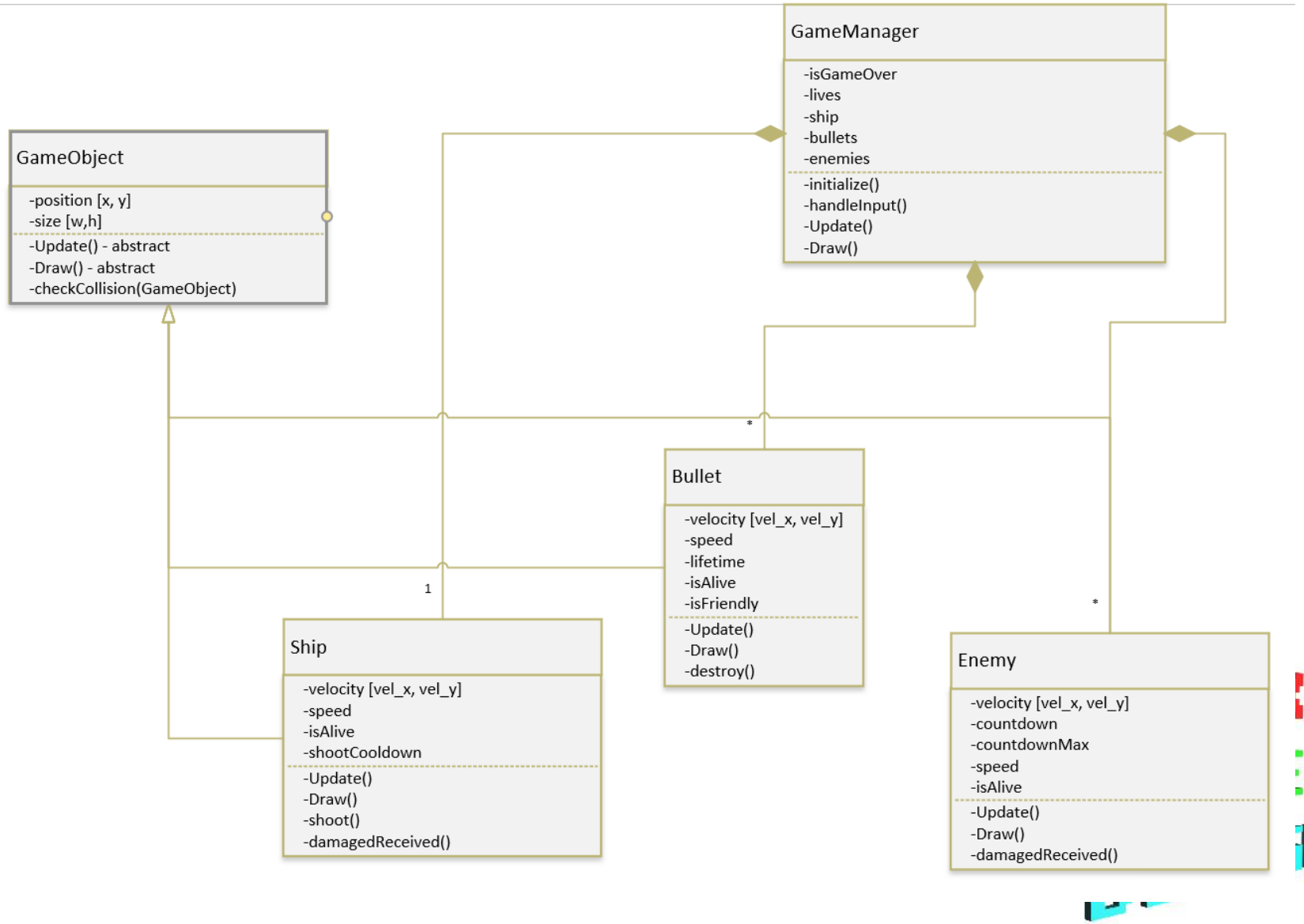
```
sprShip = al_load_bitmap("..\\ship1.pcx");
oldSchoolTransparency(sprShip);
```

```
void GameManager::oldSchoolTransparency(ALLEGRO_BITMAP *img) {
    al_set_target_bitmap(img);

    int i, j;
    ALLEGRO_COLOR c;
    ALLEGRO_COLOR c_magenta;
    ALLEGRO_COLOR c_transparent;
    c_magenta = al_map_rgb_f(1, 0, 1);
    c_transparent = al_map_rgba_f(0, 0, 0, 0);
    for (i = 0; i < al_get_bitmap_height(img); i++) {
        for (j = 0; j < al_get_bitmap_width(img); j++) {
            c = al_get_pixel(img, j, i);

            if (c.r == c_magenta.r && c.g == c_magenta.g && c.b == c_magenta.b) {
                al_put_pixel(j, i, c_transparent);
            }
        }
    }

}
```

KNOX GAME DESIGN

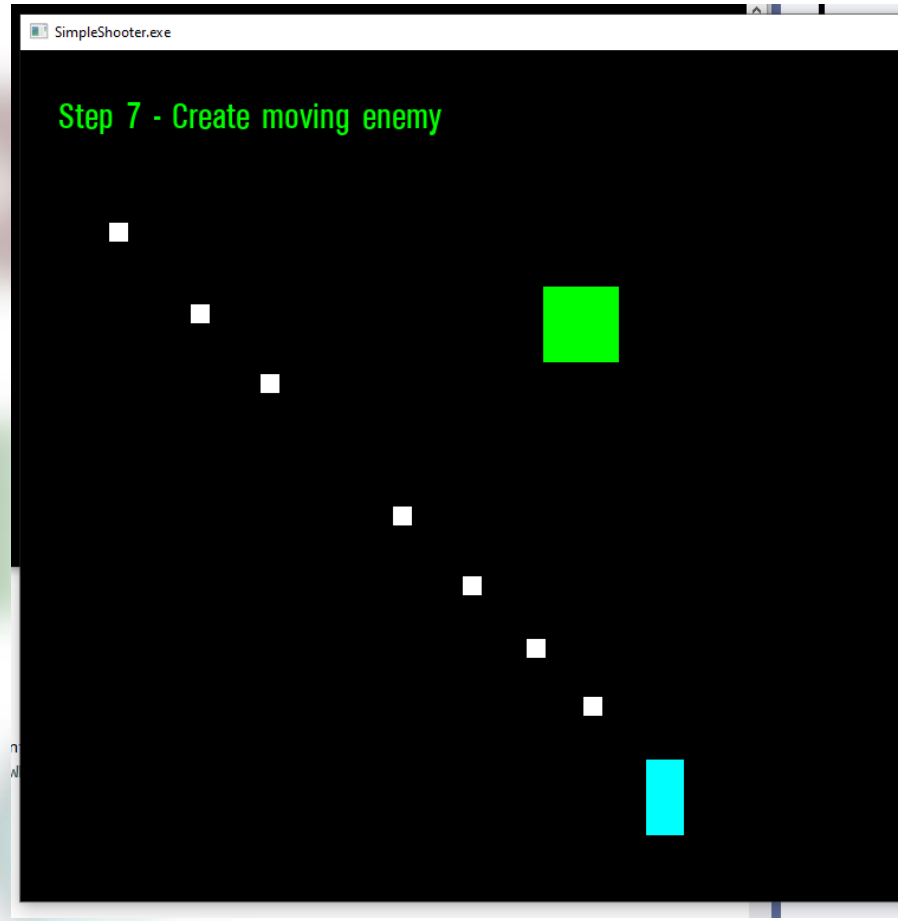
Simple Shooter

- Classes
  - GameManager
  - GameObject
  - Ship
  - Enemy
  - Bullet



Simple Shooter Steps

1. Draw ship (box) to the screen
2. Make ship move on key press
3. Check bounds
4. Stop moving on key release
5. Spawn moving bullet on key press
6. Spawn multiple bullets
7. Create moving enemy
8. Implement collision between bullet and enemy
9. Add multiple enemies
10. Collision between enemy and player



KNOX GAME DESIGN

Polishing the Simple Shooter

- 11. Game Over state; Start new game; Add restart method
- 12. Shoot cooldown
- 13. Enemy shooting; Enemy bullet/ship collision
- 14. Sound effects and Music
- 15. Add joystick controls
- 16. Add sprites, animation sequence

KNOX GAME DESIGN

References

- Official Allegro site - https://liballeg.org
- Allegro community - https://www.allegro.cc
- Allegro 5 API reference - https://www.allegro.cc/manual/5
- Delorie Software (DJGPP) - http://www.delorie.com
- DOSBox - https://www.dosbox.com
- Visual Studio - https://visualstudio.microsoft.com
- C++ reference - https://www.cplusplus.com/reference
- W3Schools C++ Tutorial/Reference - https://www.w3schools.com/cpp/
- DJ Delorie blog at Red Hat - https://developers.redhat.com/blog/author/djdelorie/

KNOX GAME DESIGN