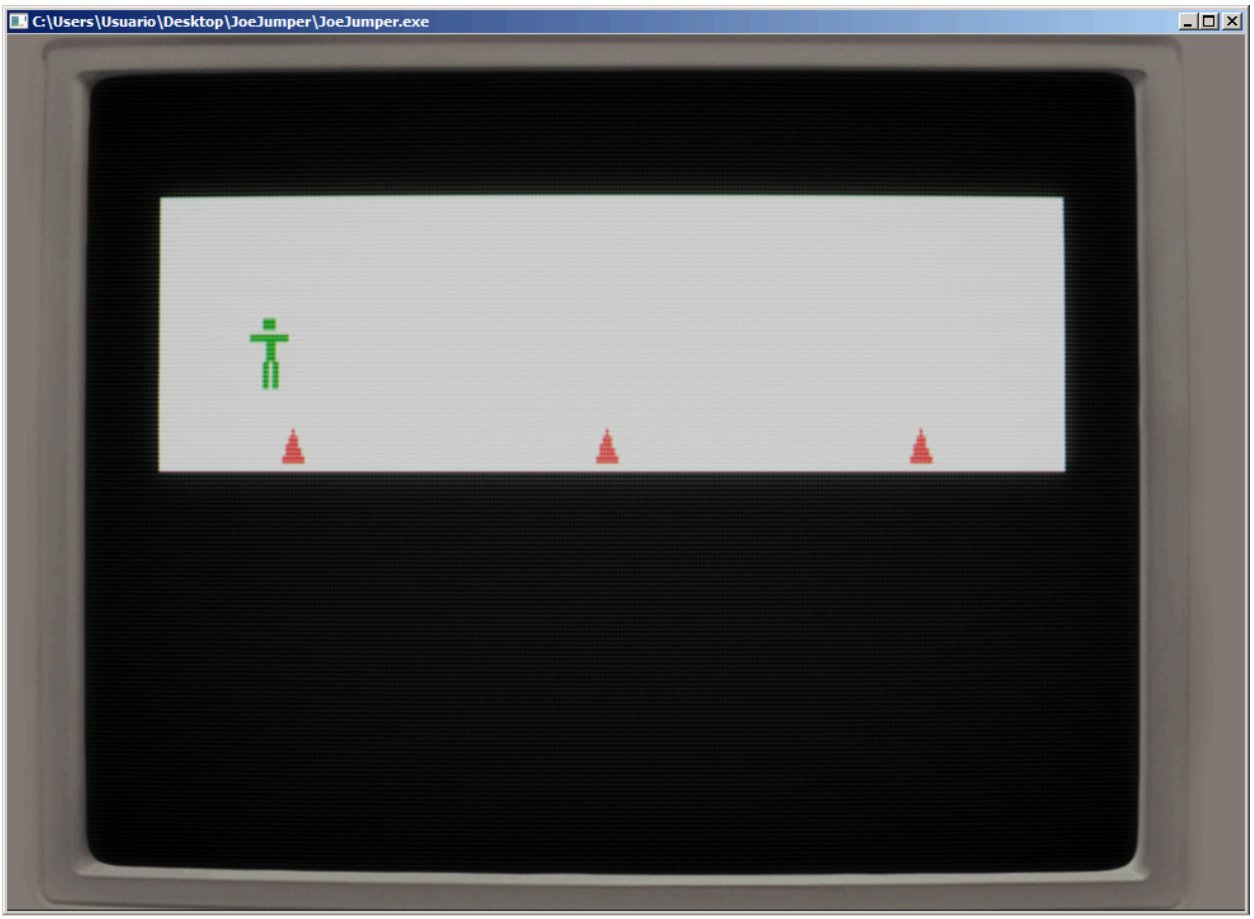
Tutorial for Beginners

This document presents a way to build a simple game with C language on windows.



You must have downloaded the project and unzipped the folder on your desktop.

First steps

You must create a file called **game.c** and you must be able to edit the source code of that file.

For now the content of this file can simply be:

```
hello world
```

To learn how to use **dos-like**, let's try to transform this code file into an **executable file**.

The **game.c** file most be in “Hello-Graphics-World (or Hello-Graphics-World-main) folder/directory.

Open the prompt and find the folder where game.c is.

The command to use the compiler is:

```
tcc\tcc game.c
```

Code in C Language

When this command is executed, the compiler throws an **'error message**'.

Because this code is invalid.

Let's edit the '**game.c**' file so that the compiler can generate an executable file.

Type it:

```
#include<stdio.h>
int main(){
      printf("hello world");
      return 0;
}
```

These five lines contain source code that the compiler understands.

If you send it to the compiler **(**with the command **'tcc\tcc game.c')** it should generate an executable file.

If it displays an 'error message' it could be a typo or prompt is not in the directory

This **'include**' line unlocks the '**printf**' function (which is used to display a message on the screen)

This line of main() tells the compiler that the code starts there.

The 'printf' line sends the '**hello world**' string to screen.

Executable File

With the command '**tcc\tcc game.c**' a file called game.exe must be generated.

This file should be executed by double-clicking for now we will use the prompt to execute it.

Type 'game.exe'

If everything is ok, a "**hello world**" message should be displayed at the prompt!

This means the 'printf' command **worked**
'include' **worked**
the 'main function' **worked**
the compiler **worked**
and the executable **worked**.

Now let's edit this source code to start programming the game.

Tiny C Compiler & dos-like

The files inside the '**dos-like**' folder are an extension of the '**tcc**' compiler

From now on, this tutorial will no longer use the printf command

Graphics Mode

The game will use **ms-dos graphics mode** to draw the elements on the screen.

The 'include stdio' line can be deleted

Let's use the include command to link the dos-like initiative code

```
#include "dos-like/dos.h"
```

The 'printf' line must be deleted.

In its place is the line that puts the screen in ms-dos graphical mode:

```
      setvideomode(videomode_320x200);
```

Inside a loop (which is closed when the user presses 'Esc') are the commands that draw on the screen

The command that chooses the light gray color to draw the game **background**:

```
            setcolor(15);
```

A variable called background stores 8 numbers corresponding to 4 **coordinates** (x,y) of the background

(20,30) = top left corner
(300,30) = top right
(300,100) = bottom right
(20,100) = bottom left corner

```
            int background[8]={20,30,300,30,300,100,20,100};
```

The fillpoly command fills the rectangle described in these 4 coordinates on the screen.

```
            fillpoly(background,4);
```

The **waitvbl** function is an important part of the dos-like library

```
#include "dos-like/dos.h"
int main(){
      setvideomode(videomode_320x200);
      while(!keystate(KEY_ESCAPE)){
            setcolor(15);
            int background[8]={20,30,300,30,300,100,20,100};
            fillpoly(background,4);
            waitvbl();
      }
      return 0;
}
```

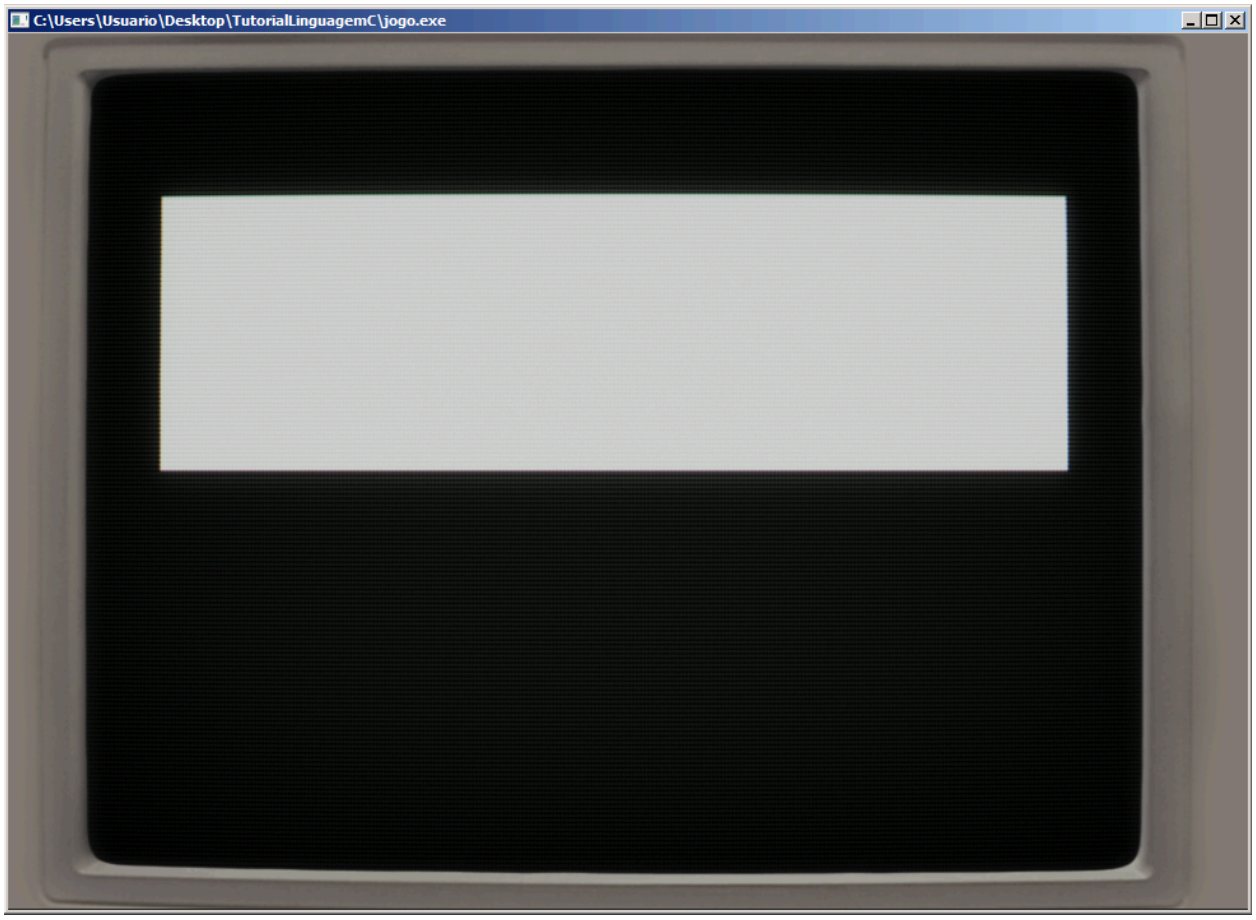
To compile (generate the executable file) the command **changed** to:

```
tcc\tcc game.c dos-like\dos.c
```

(because you need to link 'dos-like' in the executable)

Check if the prompt displays any 'error messages'

If **everything went well**, the game.exe file was generated again and it now displays a screen like this:



Drawing the Obstacle

Still within the loop, we will create a variable obs1 that will store 6 numbers.

These are the coordinates of the vertices of a triangle.

The obstacle will move around the screen, but for now it remains fixed in one position.

```
            setcolor(12);
            int obs1[6]={192,98,196,88,200,98};
            fillpoly(obs1,3);
```

Making the obstacle move

We create a variable obstacle1 with the position of the obstacle.

Then we increase the value of this variable from time to time (milliseconds)

And draw the background and the obstacle again (in the updated position)

```
#include "dos-like/dos.h"
int main(){
	int obstacle1=0;
	setvideomode(videomode_320x200);
	while(!keystate(KEY_ESCAPE)){
		setcolor(15);
		int background[8]={20,30,300,30,300,100,20,100};
		fillpoly(background,4);
		setcolor(12);
		obstacle1=(obstacle1+3)%274;
		int obs1[6]={
			292-obstacle1,98,
			296-obstacle1,88,
			300-obstacle1,98
};
		if(obstacle1>0)fillpoly(obs1,3);
		sleep(20);
		waitvbl();
	}
	return 0;
}
```

The sleep function is used to 'wait for' 20 milliseconds to pass (and then continue...)

Drawing Joe

```
		setcolor(2);
		int head[8]={52,80,56,80,56,83,52,83};
		fillpoly(head,4);
		int arms[8]={48,84,60,84,60,86,48,86};
```

```
            fillpoly(arms,4);
            int body[8]={53,86,56,86,56,91,53,91};
            fillpoly(body,4);
            int rightLeg[8]={52,91,54,91,54,98,52,98};
            fillpoly(rightLeg,4);
            int leftLeg[8]={55,91,57,91,57,98,55,98};
            fillpoly(leftLeg,4);
```

Making Joe Jump

Just like changing a variable changes the position of the obstacle, with Joe it will be the same.

The **howHigh** variable will indicate where Joe will be drawn.

The variables head, arms, body and legs will have values based on **howHigh**.

```
      int howHigh=0;
```

```
            int head[8]={
                  52,80-howHigh,
                  56,80-howHigh,
                  56,83-howHigh,
                  52,83-howHigh};
            fillpoly(head,4);
            int arms[8]={
                  48,84-howHigh,
                  60,84-howHigh,
                  60,86-howHigh,
                  48,86-howHigh};
            fillpoly(arms,4);
            int body[8]={
                  53,86-howHigh,
                  56,86-howHigh,
                  56,91-howHigh,
                  53,91-howHigh};
            fillpoly(body,4);
            int rightLeg[8]={
                  52,91-howHigh,
                  54,91-howHigh,
```

```
                54,98-howHigh,
                52,98-howHigh};
            fillpoly(rightLeg,4);
            int leftLeg[8]={
                55,91-howHigh,
                57,91-howHigh,
                57,98-howHigh,
                55,98-howHigh};
            fillpoly(leftLeg,4);
```

To **handle** jump, 5 **conditions** will be necessary:

if Joe is on the ground and starts jumping
if joe is rising
if joe is coming down
the moment it stops rising
the moment he stops descending

```
            if(howHigh==0&&keystate(KEY_UP))howHigh=5;
            if(howHigh%4==1)howHigh+=4;
            if(howHigh%4==3)howHigh-=4;
            if(howHigh>28)howHigh=31;
            if(howHigh<2)howHigh=0;
```

Finishing

In the final version, collision detection and a 'phase restart' are implemented:

```
#include "dos-like/dos.h"
int obstacle1,obstacle2,obstacle3,howHigh,gameover;
void init(){
      obstacle1=-250;
      obstacle2=-250;
      obstacle3=-250;
      howHigh=0;
      gameover=0;
}
int main(){
      setvideomode(videomode_320x200);
```

```
	init();
	while(!keystate(KEY_ESCAPE)){
		if(gameover!=0){
			if(keystate(KEY_UP))init();
		}else{
			if(howHigh==0&&keystate(KEY_UP))howHigh=5;
			if(howHigh%4==1)howHigh+=4;
			if(howHigh%4==3)howHigh-=4;
			if(howHigh>28)howHigh=31;
			if(howHigh<2)howHigh=0;

			setcolor(15);


			int background[8]={20,30,300,30,300,100,20,100};
			fillpoly(background,4);
			setcolor(2);
			int head[8]={
				52,80-howHigh,
				56,80-howHigh,
				56,83-howHigh,
				52,83-howHigh};
			fillpoly(head,4);
			int arms[8]={
				48,84-howHigh,
				60,84-howHigh,
				60,86-howHigh,
				48,86-howHigh};
			fillpoly(arms,4);
			int body[8]={
				53,86-howHigh,
				56,86-howHigh,
				56,91-howHigh,
				53,91-howHigh};
			fillpoly(body,4);
			int rightLeg[8]={
				52,91-howHigh,
				54,91-howHigh,
				54,98-howHigh,
				52,98-howHigh};
			fillpoly(rightLeg,4);
			int leftLeg[8]={
				55,91-howHigh,
				57,91-howHigh,
				57,98-howHigh,
				55,98-howHigh};
```

```
                    fillpoly(leftLeg,4);
                    setcolor(12);
                    obstacle1=(obstacle1+3)%274;
                    obstacle2=(obstacle2+4)%274;
                    obstacle3=(obstacle3+5)%274;
                    int obs1[6]={
                          292-obstacle1,98,
                          296-obstacle1,88,
                          300-obstacle1,98};
                    int obs2[6]={
                          292-obstacle2,98,
                          296-obstacle2,88,
                          300-obstacle2,98};
                    int obs3[6]={
                          292-obstacle3,98,
                          296-obstacle3,88,
                          300-obstacle3,98};
                    if(obstacle1>0)fillpoly(obs1,3);
                    if(obstacle2>0)fillpoly(obs2,3);
                    if(obstacle3>0)fillpoly(obs3,3);

if(obstacle1>232&&obstacle1<242&&howHigh<9)gameover=1;

if(obstacle2>232&&obstacle2<242&&howHigh<9)gameover=1;

if(obstacle3>232&&obstacle3<242&&howHigh<9)gameover=1;
                    sleep(20);
                    waitvbl();
               }
          }
          return 0;
}
```