REAL TIME SYSTEMS FINAL PROJECT

TITLE: SHOOTING

Student: Valdo João
ID: 507104
Contact: valdojoao90@gmail.com

Date: 22 / 01 / 2014

Contents

1. Introduction

This document report shows the design and implementation of the Shooting computer game which was developed under Linux operating system and C programming language, using pthread library for multithreading and allegro library for the graphics design. The game is intended for anyone to play in order to test his capabilities of shooting flying pigeons on the screen.

The description of the project, what it looks like and how it works is thoroughly discussed. The user interface of the project is also presented. The goal of the game is to generate pigeons on the screen which are capable of flying on random positions with a certain speed and the player tries to shoot them with the help of a target.

2. Game overview

The main goal of this project is to know how to manage multiple threads. Within this project there are different tasks which are managed by different threads in order for the game to work accurately. This section elaborates on the description of the game and users interface.

2.1 General Description of the project

The game can be divided into two main components which can be classified as are *characters* and *environment*.

A. Characters: there are two kind of characters which are:

i. The Target

The major character of this game which represents the player is the target. The target image is visible on the screen and the player can move it with the mouse. The player will move the target image on the screen in order to try to match the current position of a certain pigeon with the current position of target center.

In order to shoot the aforementioned pigeon, the player must press the left button of the mouse. The player can only shoot if he has enough bullets to do so, otherwise the gun is empty and he must press the key “R” in order to reload it.

ii. Pigeons

The Pigeons will start flying on the forest, with a certain speed, as soon as the game is started. When the game is started, the first position of each Pigeon is determined automatically by a random function. After their first appearance each Pigeon's next position in the forest will be decided again by a random function. Six Pigeons will be generated on the screen. They will be divided in groups of two according to the speed which are slow, medium and fast.

If all the Pigeons flying on the screen are shot successfully and there is still time available to play, another set of Pigeons will be automatically generated, and a bonus will be given to the player which consists of increasing the number of times he can reload the gun.

B. Environment: there are different environments in the game window such as:

i. Time: this panel shows how much time is left to play the game, the player can set a new game by pressing the key "N".

ii. Score: in this panel is shown the points earned by the player.

iii. Bullets: this panel shows how many bullets are available to shoot.

iv. Reload: this panel shows how many times the player can reload the gun.

2.2 Design choices

The design choices of this project where made under the following circumstances:

i. Real-Time policies: in this case was chosen SCHED_FIFO (Fixed-Priority Scheduling + FIFO). Under this policy the threads with the same priority are handled by a FIFO policy. A thread will be executed until termination, cancellation, blocking, or preemption.

ii. Linux scheduler: In this situation was chosen pthread_attr_setinheritsched for a thread to be scheduled with different policies used by the father.

iii. Timing constraints: for timing constraints it's used CLOCK_MONOTONIC which represents the time elapsed from a given undefined instant of time. It is not affected

by adjustments, hence it is the best solution to measure the time elapsed between two events.

iv. **Semaphores:** for the use of mutual exclusion semaphores, was chosen Mutex which can only be used for mutual exclusion, not for synchronization, a Mutex locked by a thread can be unlocked only by the same thread. Mutual exclusion regulates the use of shared resources so that tasks can only access them one at the time.

2.3 User Interface

There are three interfaces that the player interacts with.

i. **Welcome user Interface:** In this interface the player is welcomed to the game and he has two options. Quit by pressing “ESC” or start the game by pressing “ENTER”.

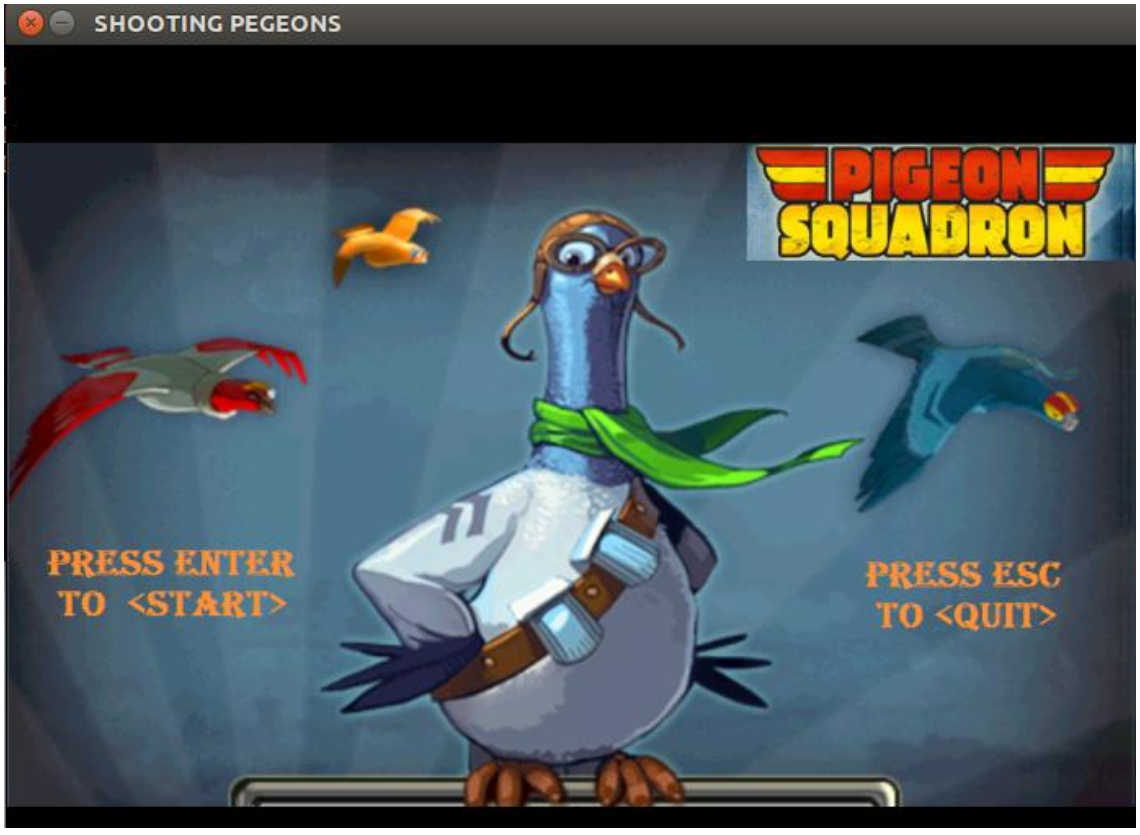

Figure 1: Welcome User Interface

ii. **Game Playing user Interface:** User interface where the player plays the game.

Figure 2: Game Playing User Interface

iii. **Game Over user Interface:** this is the interface where the game ends.

Figure 3: Game over user Interface

3. Shared data structures

The data structures defined in the project are:

i. **task_par:** defines the real time parameters for each thread.

```
/* real time parameters for each thread */
struct task_par {
        int arg;                                /* task argument */
        long wcet;                              /* in microseconds */
        int period;                             /* in milliseconds */
        int deadline;                           /* relative (ms) */
        int priority;                           /* in [0,99] */
        int dmiss;                              /* no. of misses */
        struct timespec at;                     /* next activ. time */
        struct timespec dl;                     /* abs deadline */
};
```

Figure 4: task_par

ii. **pig:** composed by two tasks which defines the pig's parameters

```
typedef struct _pig_
{
  int x,y;                              /* x and y position of the pig */
  int id;                               /* pig id */
  BITMAP *pic;                          /* pig bitmap*/
} pig_;

pig_ pig[NO_OF_PIGS];


typedef struct _pigLife_
{
  int state;                            /* state of the pig, 1 alive and 0 dead */
  int dir;                              /* direction of the pig */
  double speed;                         /* speed of each pig */
} pigLife_;

pigLife_ pigLife[NO_OF_PIGS];
```

Figure 5: pigs par

4. Task's description

Each task of this project was scheduled by Linux real time policy, SCHED_FIFO scheduler in which threads with the same priority are handled by a FIFO policy. A thread is executed until termination, cancellation, blocking or preemption. The tasks (threads) of the game are the following:

i. **target_img_moving:** this task is responsible for moving the target image around the screen according with the mouse coordinates. The time constraint of this task is 20ms.

ii. **countdown:** this the task checks if the countdown time reaches the next second.

iii. **movepig:** this task is responsible for moving the pigeons which are alive around the screen. In order to do that, a random direction will be generated with a fixed speed to reach the desired location. The time constraint of this task is 20ms.

iv. **shootpig:** this task is responsible to shoot when the left button of the mouse is pressed. It also responsible for checking if there are enough bullets to do so. This task is also responsible to play all the sounds of the game, that's it, when the gun is shot with bullets, when the gun is shot with empty bullets and when the gun is reloaded. If a Pigeon is successfully shot this task set his state to dead and increases the score. The time constraint for this task is 20ms.

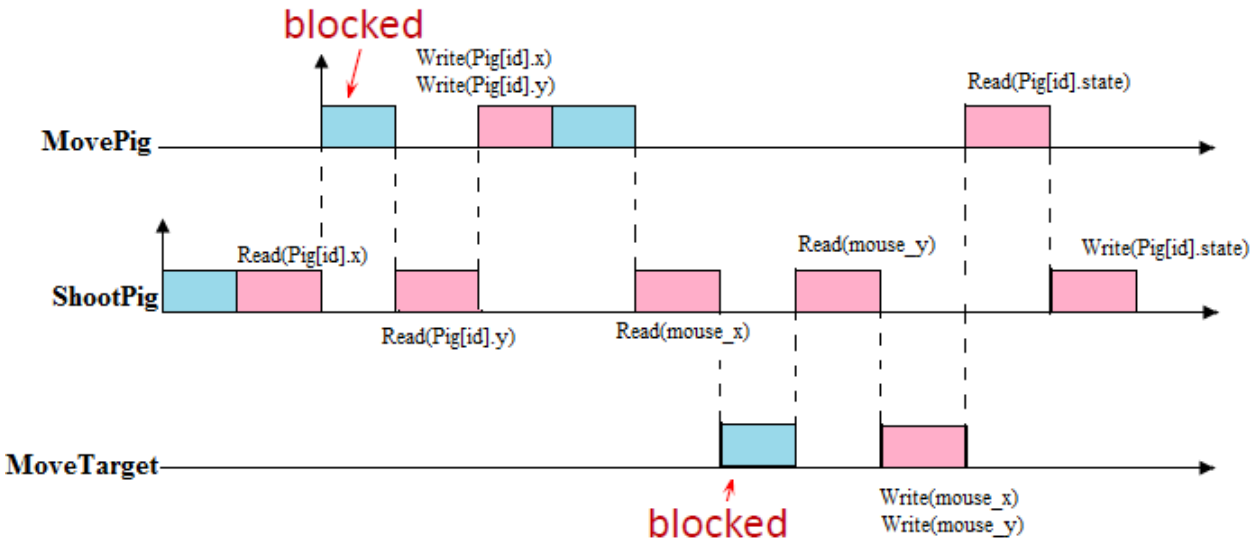

Figure 6: interactions between tasks and resources

5. Conclusion

The main goal of the project, which was knowing how to manage multiple threads, was accomplished. The project also tried to provide a user friendly application with some instructions in order to enhance the player's experience. Nevertheless, there are some improvements to be done such as adding levels to the game where the speed of the pigeons will be increased upon each level termination and play a sound when a pigeon is shot successfully.

6. References

- Pearson Education; Game Programming and Graphics in C with Allegro; 2007
- BALDWIN, Richard; Graphics Programming using Allegro; http://www.dickbaldwin.com/allegro/Allegro00120.htm, 2008
- LEVERTON, Matthew; Alegro cc; https://www.allegro.cc/manual/4/
- HARBOUR, Jonathan; Game Programming All in One; 2007