Instruction; project number 2

Basics Of Computer Programming 2020/21, Data Engineering

author: Robert Ostrowski[1]

version from: 15.11.2020 r.

Project Robot Unicorn Attack

The goal

The goal of the project is to implement the game „Robot Unicorn Attack” and to fulfill ones wishes (about passing the project). The essence of the game is to control jumps and dashes of a unicorn. The unicorn is permanently running in one direction. The goal is to maximize the number of points, by staying alive as long as possible. The chosen functionalities that has to be implemented are listed below. The basic version is simplified, comparing to the original game. The elements of a level are simplified to rectangle shapes. The surfaces on which the unicorn moves are simplified to the flat ones.

The description of the game can be found at:
https://en.wikipedia.org/wiki/Robot_Unicorn_Attack
https://gamefaqs.gamespot.com/flash/996571-robot-unicorn-attack/faqs/59954
The game is available here:
https://unicorn.jocke.no/ [warning, the flash included]

Programming environment

To the instruction there is attached a basic program which includes:

- Calculation of time increases, which allows to measure the times spend
- Drawing bmp graphic files
- Drawing a pixel, a line, and a rectangle
- Displaying a text

The program uses SDL2 (2.0.3) – http://www.libsdl.org/ library. It is included in the basic project – there is no need to download the sources.
The compilation under a 32-bit Linux system can be done be the following command:

```
g++ -O2 -I./sdl/include -L. -o main main.cpp -lm -lSDL2 -lpthread -ldl -lrt
```

and under a 64-bit system with the following command:

```
g++ -O2 -I./sdl/include -L. -o main main.cpp -lm -lSDL2-64 -lpthread -ldl -lrt
```

To compile the project, the directory containing main.cpp shall contain:

- Bitmaps with the needed pictures (cs8x8.bmp, eti.bmp), be sure to preserver proper

1 In the case of doubts or inconsistencies in the instruction please contact the author. The consultations are on Wednesday 6pm-8pm. The link is at enauczanie.

capitalization of letters.
- The libSDL2.a file (libSDL2-64.a when compiling a 64-bit version)
- The sdl directory attached to the project.

To the project, there are attached scripts that can be used for a compilation (comp for a 32-bit version environment and comp64 for a 64-bit version environment).

The presentation (for the grading) of the project will be in the operating system chosen by a student. The following choices are available:

- Linux system: The student is required to check if the project can be properly compiled and runs correctly under the laboratory distribution.
- Windows System and compiled using MS Visual C++ studio with the version available at the laboratory.

The successful execution of the program during the presentation is required to obtain points from Project 2.

The program shall not use the C++ stl library.

Obligatory functionality (5 pts)

All the functions have to be implemented. The version of the game implementing the following functions is simplified comparing to the original game. In particular the controls can be used to test consecutive functions. Lack of any of the following functions is equivalent with obtaining 0 points from the project.

1. Basic GUI: border of the board, space for printing the extra information (time elapsed from start of the level).
   Implementation of keyboard controls:
   a. *Esc:* exit the program
   b. *n*: new game

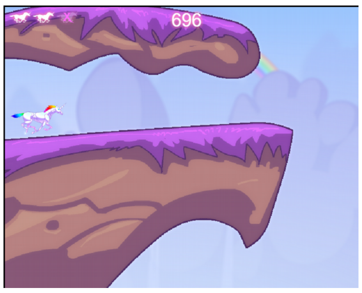
2. Keyboard controls for moving the unicorn: the arrows keys. The position of the unicorn shall be bounded by the resolution of the game, not by the layout of the blocks of which the board is build off. The „physics” of jumps is not required. The unicorn shall be always in the left lower corner of the board.

3. An implementation of one level of the game. The level shall contain obstacles which the unicorn shall evade (the collision of the unicorn with the obstacles shall be implemented). The level shall be at least a few times wider than the width of the board. It should be possible to move on the level. The active – visible to the player - portion of the level shall be correctly displayed. The level shall be looped, in X dimension. That is, after moving from a point into some direction for some time the same position shall be obtained.

4. There is no need to implement end of a level (i.e. that the unicorn died), but the elapsed time in the level shall be measured. The level shall contain a floor everywhere - it is not possible for the unicorn to fall by the floor. All the obstacles are rectangles „laying” on the floor. There are no “platforms” hanging above the ground.

Extra functionalities (10 pts.)

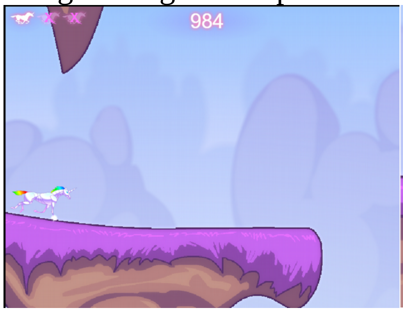
1. **(1 pkt)** Implementation of default unicorn controls. The controls replace the basic controls from point 2 of obligatory functionalities.
   a. The change of control mode shall be done after pressing **‘d’** key.
   b. The unicorn moves in the right direction by default, gaining speed with time.
   c. ‘**z**’ - jump. During the jump the unicorn can dash or make another jump (double jump). Pressing the button for longer increases the height and range of the jump till some fixed value. The unicorn regains the full ability to jump after touching a ground.
   d. **‘x’ -** dash. The unicorn can do a quick horizontal move. During the dash, the unicorn do not loss height but also can not do anything else. However, ending the dash allows the unicorn to make another jump, even if it used double jump already.
   e. The parameters used for the unicorn move shall be easy to change, to obtain smooth move.

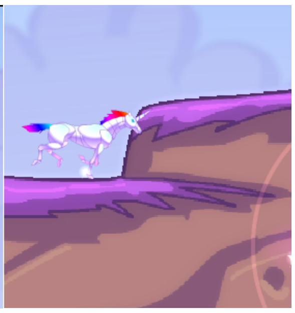


2. Advanced visualization (horizontal platforms):
   a. The level consist of hanging platforms on which the unicorn can jump on and move. To allows this, the collision detection shall be implemented: the unicorn moving on the platform does not fall through it.
   b. The level is not only wider than the width of the board but it is also higher than the height of the board. The unicorn is positioned in the middle of board provided that it is not close to the lower and upper limit of board. The sample level shall contain a few platforms at different heights and at least one platform that is above another one.
   c. The obstacles from subpoint 4 of the obligatory functionalities can be positioned on the platforms.
   d. **Attention, a simplification, comparing to the original game:** If the bonus functionalities are not implemented, then the level may consist of rectangular blocks, and all surfaces for movement can be flat and horizontal

3. **(1 pt)** Obstacles and collisions.
   a. The obstacles on platforms in the form of terrain of greater height.
   b. Addition of stalactite-like obstacles. The obstacles are not a part of ground. The collisions with such objects have to be correctly implemented. In particular the unicorn cannot pass them.

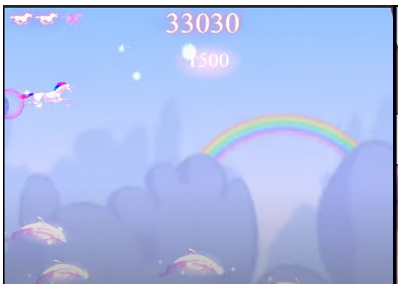
c. **Attention, a simplification:** The shapes of the obstacles can be approximated using rectangular shapes of which the level consists.

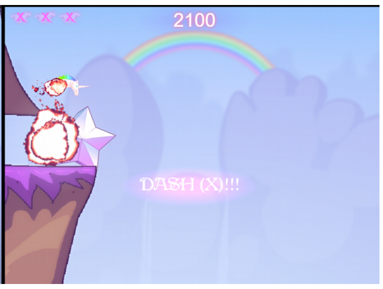




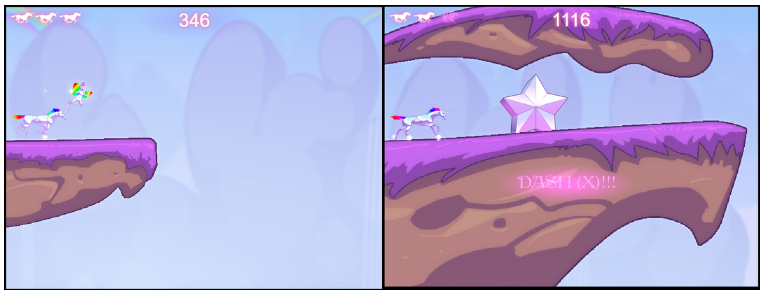
4. **(1 pt)** Death, the number of lives and the menu:
    a. Displaying the remaining number of lives of the unicorn in a graphic form.
    b. A loss of a life after a head-on collision with the terrain.
    c. Falling off a platform and falling off the level. The level does not contain a continues ground.
    d. A collision with a star without the dash. A collision during the dash is safe – the unicorn passes the star.
    e. After losing a life a prompt shall be displayed. The prompt shall ask if to continue and present, if Point 6. was implemented, the number of points obtained.
    f. Losing all the lives shall result in displaying the main menu. Or, if the point 8. was implemented, there shall be an option to save the score obtained.

5. **(2 pts)** Animations (some subpoints require to implement other functionalities):
    a. An animation of the movement, jump and falling of the unicorn.
    b. An animation of a star explosion, when the unicorn collides with it during the dash. An animation of destruction of the unicorn in the case of collision with the star.
    c. The bonus points obtained shall be briefly displayed on the screen.
    d. Dolphins jumping from the lower board border – after the player passes milestones the dolphins are coming to admire the progress of the player.
    e. **Attention:** the speed of the animation shall be independent of the computer speed (provided that it fulfills the minimum requirements of the game).





6. **(1 pt)** Increasing the points.
    a. The points shall be increased constantly, proportionally to the distance passed by the unicorn.
    b. For breaking the stars: the stars are the obstacles that can be destroyed after a collision during the dash. 100 points are awarded for the first star and 100 more for each next. Omitting the star results in a reset of the counter, i.e. the next star broken is again worth only 100 points.
    c. For collecting the fairies. The fairies are figures moving randomly in a given space, a collision with a fairy is equivalent with collecting it. 10 points are awarded for the first fairy, 10 points for each next. An omission of a fairy resets the extra points for a fairy to 10.



    d. The fairies and stars shall appear with some probability in the places depending on the level. If the level loops the position of these extras shall be not always the same.
    e. The number of obtained points shall be displayed on the board.

7. **(1 pt)** Saving the best results:
    a. After end of the game it should be possible to add a nick and the number of points associated with the ended game to a file.
    b. It should be possible to view the list of sorted results, as an option in the main menu.
    c. The number of results on the screen shall be bounded.
    d. If there is not enough space on the screen to display all the results, the program shall allow to view the results by scrolling a list of the results or by allowing to change pages.

8. **(1 pt)** Encoding a level in a file.
    a. An individual file format to store an information about a level shall be developed. The files should be editable – for example in a notepad. The student shall be familiarized with the format enough to discuss it and edit a level during the presentation. The program shall be able to load the description of a level from the file. The file shall contain the information about the position of all of the obstacles and the size of level, at least.
    b. In addition, the file shall contain the information about all extra functionalities

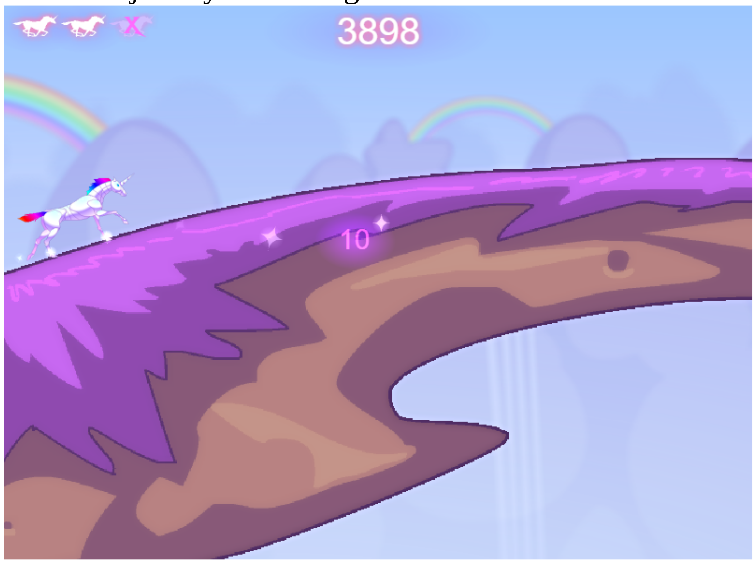
implemented.

c. **Attention:** The program shall not limit the maximum number of objects in a file. It means that the program shall be able to analyze the content of the file and allocate the memory to store the representation of all the objects contained in the file. The encoding of this information shall be developed individually. It means, in particular, that a kind of preamble can be added to the file, to describe the objects contained in the level, and the size of the level. This allows to read the preamble, allocate memory according to the preamble, and then to load a level. The consistency of the preamble and the data shall be checked.

Bonus functionalities (3 pts)

Attention: the bonus functionalities are extending the previous ones: i.e. the functionalities about composition of a level, collisions, encoding of a level, etc. **Hence, they are graded only if all extra functionalities are implemented.**

1. **(1.5 pt)** Even more complex levels (complex shapes of platforms)
   a. The elements of which the level consists can have irregular shape, independent of provided bitmaps. In particular the surfaces of a platforms on which the unicorn moves can be described by function. To find suitable functions one can use https://mycurvefit.com/, for example; and use a polynomial representation of the function.
   b. If during the dash the unicorn moves onto a surface with increasing height then it continues to move on this surface (not horizontally). The dash on a surface of diminishing height make the unicorn lose the contact with floor.
   c. The polygon representing the platform can be only an approximation of real trajectory of moving unicorn.



1. **(1.5 pt)** Dynamic construction of a level:
   a. The shapes of available platforms should be stored in the file describing a level.

b. If the part of the level ends in a gap, the position of the next platform shall be generated in a way that the unicorn should be able to jump onto this platform. The possible positions shall be determined by the rules of movement of the unicorn not by skills of a player.

Final observations:

- Requirements about graphics: it is enough to use any bitmaps (with a proper sizes).
- The configuration of the program shall allow to change all the parameters easily, not only these considered in this instruction. The easy change means changing a parameter in the program.
- The project can be written using objects, but completely forbidden is to use standard C++ library (in particular string class, cin, cout, vectors, etc.). You can use the content of string.h library from C std language. Do not confuse the content of this library with operations on string type in C++.
- The files shall be processed using standard C library – the family of f* functions (like fopen, fread, fclose, etc.).
- Every part of the coded submitted for the grading shall be realized individually.
- The speed of the game shall be independent of the speed of the computer on which the program is run.

  The constant parameter in the program shall be described using suitable comments, for example:

```
const int WIDTH_OF_BOARD = 320;    // pixels
const double XPOSITION_OF_TEXT = 60.0; // % of the screen width
const double YPOSITION_OF_TEXT = 5.0;  // % of the screen height
```