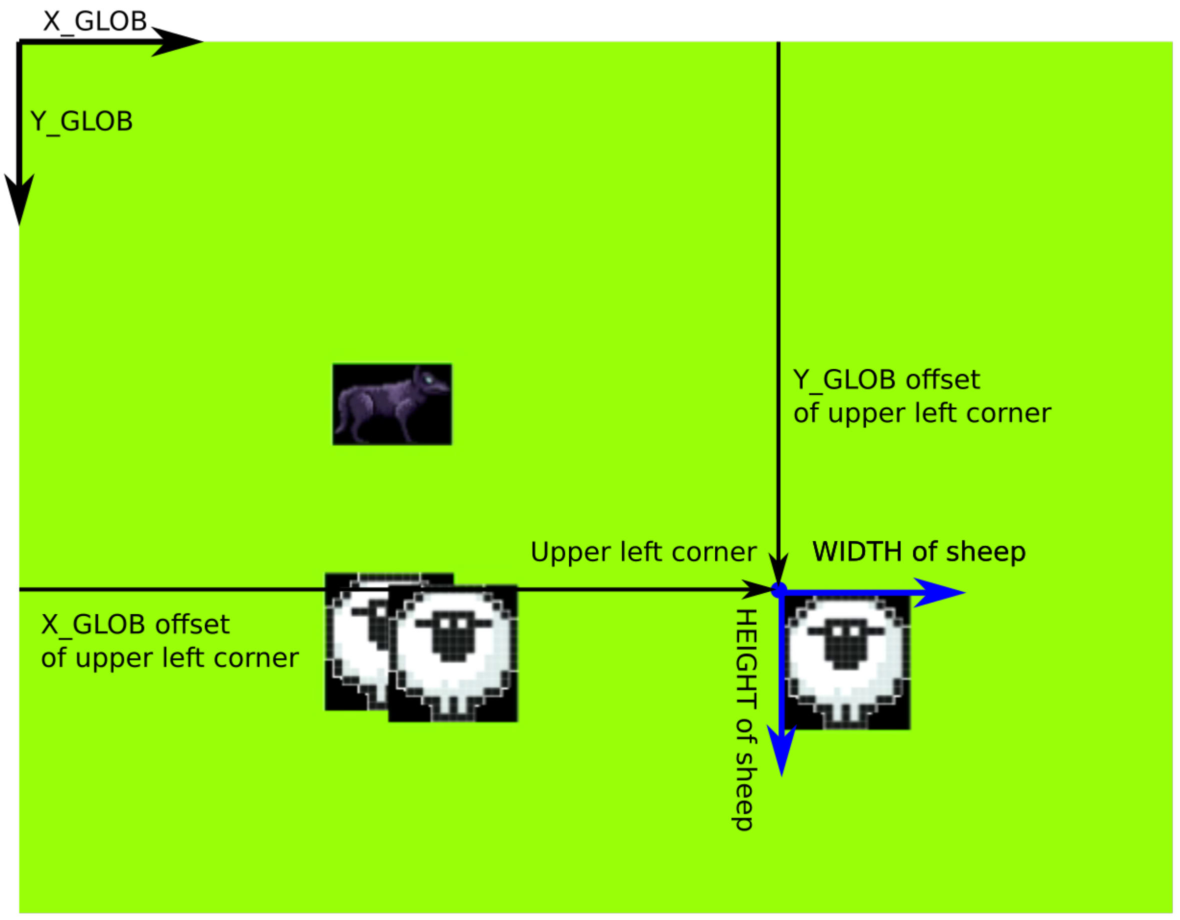

Figure 1: Screenshot from example execution

1 SDL Project Short Intro

The overall goal of the project is to create a *game* in which multiple species of animals exist and proliferate. In the final version, we also want to add a controllable character. For instance one could go for sheep, wolves, shepherd dogs and a shepherd as playable character. These animals will then move around, interact with one another and also be influenced by the controlled character. For visualization we will use the SDL2 libreary, see `https://www.libsdl.org`.

2 SDL Project Part 1

The goal of the first part of the project is to get to know the library and have a first relatively simple application in the end. We want to have a window on which different animals spawn randomly, walk around for a short period of time and then redisappear.

The window created by the application should be similar to fig. 1. (Obviously without the arrows and annotations.)

First we will introduce the functionalities of SDL which we are going to use, before sketching the *skeleton* of the classes involved.

2.1 SDL API

- **SDL_Rect** Defines a rectangle via its upper left (!) corner as well as its width and height. We will mainly use this to position textures.
- **SDL_Surface** Basically a rectangular array of pixels representing the screen or a texture. Note that loading a surface in SDL (using **IMG_Load**) for instance, returns an OWNING pointer to a surface. This means we need to ensure deletion by "hand" using **SDL_FreeSurface**.
- **SDL_CreateWindow** Creates the actual "window" of the application. A pointer to the corresponding surface is obtained via **SDL_GetWindowSurface**. Note that before quitting your program, the surface needs to be freed and the window closed (**SDL_DestroyWindow**), in this order.
- **SDL_FillRect** Fills a surface with a single color. Called to clear the screen before drawing each object.
- **SDL_BlitScaled** Scales and translates a texture to match a given rectangle and "draw" onto the given surface.
- **SDL_UpdateWindowSurface** Updates what is actually shown on the screen.
- **SDL_GetTicks** and **SDL_Delay** Can be used to control your framerate. **SDL_GetTicks** returns the time passed since initialization in milliseconds, **SDL_Delay(x)** blocks the execution for **x** milliseconds.

3 Needed classes and their structure

- **animal** a base class for all animals. As all animals need to be drawn onto the screen, they need a texture (SDL_Surface loaded from image), a position (in x and y), a method draw, that allows to draw them to a given screen, a finally virtual method defining the movement.
- **sheep, wolf etc** derived from animal. Needs to load the correct texture, choose an initial position, destroy the texture on destruction and has to implement the movement function. Note that for the first part, sheep can simply follow straight lines and bounce of the edge of the window (Like the old windows screen saver); wolves move randomly or go to the nearest sheep (bonus)
- A ground object **ground** representing the "arena". Members: A counter (of type unsigned) defining the framerate.
  Needs a constructor, destructor and a function called loop() doing the actual work (It corresponds to the "main" loop of your program). Members: **ground** Holds a NON-OWNING ptr to the screen (See main application). A vector of (smart?) pointers with all the animals currently existing.
- **application** The main application. Members: The "actual" application **SDL_Window**. Its surface, **SDL_Surface**, defining what is drawn on the screen. Finally a ground object, defining the arena. Needs to call the loop() function of ground and update the

screen. In this first part we want to stop the execution after a given period of time (input in seconds).