Blobs:

1. Blobs simulation:

It is desirable to simulate the behavior of a group of Blobs that interact with each other and must follow several rules. The process starts with several blobs entered by the user in the graphical user interface.

a. Tick t = 0. The blobs start in their original positions, with their respective initial directions.
b. Tick t = 1. Blobs trace a radius of smellRadius units and check if there is some food inside that radio. If it founds food, the blob modifies its direction to get the closest food in a straight line. This operation is repeated to all blobs and then their move in the same direction that they have until they find food.
c. Tick t = 2. The blobs repeat the b. step.
d. Space is a toroidal space, so if a blob goes outside the borders of the map, it appears in the opposite limit. For instance, if the blob moves one unit to the left and there are no more possibilities to moves to the left, it continues its path appearing in the right part of the screen.
e. The simulation continues until the user decides to turn it off.
f. Blobs characteristics:
   i. The blobs have a position and a movement direction and each time the blob must modify its direction in that direction.
   ii. The blobs have speed: The program must be able to run in two different working modes:
      - In mode 1 the blobs must use the same speed. The user sets the speed with 2 values:
         - Maximum speed measured in px/tick (chosen before the simulation starts)
         - **Percentual speed** between 0% and 100% (a simulation parameter)
         - The total speed is the maximum speed multiplied by the percentual speed.

- In mode number 2, each blob has a maximum speed different from the others, those are chosen randomly between 0 and a maximum value chosen by the user before the simulation starts. Every blob shares the percentual speed.

iii. The initial directions of the blobs are calculated randomly. Then, during the simulation, the direction changes due to various phenomena:

- **BlobFeeding**: follow the b point explained before, when the blob detects that food is at a radial distance less than smellRadius, modify its direction to move towards the food
- **BlobMerg:** The details of this phenomena are included in the iv-item below.

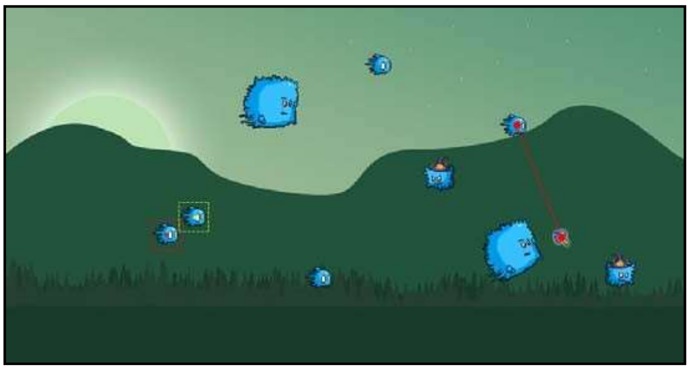
iv. The life cycle of Blobs has three age groups:

BabyBlob	**GrownBlob**	**GoodOldBlob**

- They are born in the first state which is named **BabyBlob.**
- When collide two or more **BabyBlob**, they come together to form a **GrownBlob** (They are considered to "collide" when their bitmaps overlap).
- In the same way, the collide of 2 or more GrownBlobs make them to come together to form a **GoodOldBlob**
- The fusion process is named **BlobMerge**
- As a result of a **BlobMerge**, the new direction is calculated as the sum of the average of the direction of the blobs in the screen and a random number between 0 and **randomJiggleLimit**, which is a simulation parameter between 0 and 360, which is details in the next lines.
- The resultant speed is the average of the speeds of every blob.
- When a collision is not from blobs in the same age groups, or when they can not evolve because they are **GoodOldBlob**, the Blobs can cross without interfering in their directions.

g. In every moment must be some foods in the screen defined by a parameter entered by the user which is called **foodCount**, the food appears randomly on the screen. When the blob gets the food, internally count the number of foods ate, this process is named **BlobFeeding**
h. when a certain amount of food is reached, the miracle of the BlobBirth occurs, a new BabyBlob appears on the screen in a random position and the foodCount of the parent of the blob turn to cero again.
i. The amount of food necessary to the BlobBirth is 5, 4, 3 according to **BabyBlob**, **GrownBlob, and GoodOldBlob.**
j. At every tick the BlobDeath can appear, the unexpected end of a blob life, every blob has one Death probability per tick. At the beginning of every tick, for every blob, a random number between 0 and 1 is chosen. If the number is less than the death probability of the blob, the blob dies. Every blob in the same age group has the same death probability. Those three parameters are numbers that the user can modify while the simulation is running.

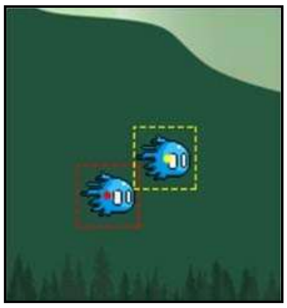
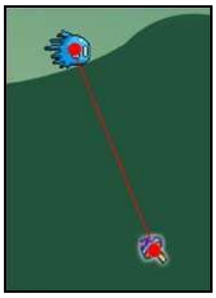
Graphic User Interface



a. The chair provides sprites for each of the Blobs, the food and the simulation wallpaper.

b. The simulation has parameters that must be able to be modified before and during the simulation, with the exceptions indicated. They are as follows:

   a. Simulation mode: 1 or 2 (must not be modified during simulation).
   b. Initial amount of blobs (should not be modified during simulation).
   c. Maximum speed of blobs. It changes its operation according to whether it is mode 1 or mode 2.
   d. Percentage speed of the blobs. (it is suggested to use a slider)
   e. smellRadius: Food detection radius from the center of the Blob.
   f. randomJiggleLimit: Random value to add to calculate the direction of the Blob.
   g. Probability of death for each age group ϵ(0,1)
   h. foodCount: Amount of food.

	
For collisions, blobs are not considered on-time. There is a collision when the bitmaps of two Blobs of the same age group overlap. The figure above shows two blobs with the border of their bitmaps in a dotted line. As they overlap, they are colliding.	Blobs and food are considered on time for all cases except collisions. The distance between two elements (Blob-Blob or Blob-Food) is measured between the central positions of both elements, and not from the edge of their bitmaps!