Bird Flocks Simulation
Documentation

Anna Danot, Núria Fernández,
Jan Mousavi, Loredana Sandu

Jun 07, 2021

CONTENTS

CHAPTER

ONE

PARAMETERS MODULE

Parameters used while running the simulation.

DIM:
(*int*) dimension of the container and simulation (2 for plane, 3 for cube).

NUM_BIRDS:
(*int*) number of birds in simulation.

ATTRACTION_POINTS:
(*list*) coordinates where attraction points will be initially located.

REPULSION_POINTS:
(*list*) of coordinates where repulsion points will be initially located.

W_AVOIDANCE:
(*float*) ratio of importance of rule of avoidance over the rest.

W_CENTER:
(*float*) ratio of importance of rule of center over the rest.

W_COPY:
(*float*) ratio of importance of rule of copy over the rest.

W_VIEW:
(*float*) ratio of importance of rule of view over the rest.

W_ATTRACTION:
(*float*) ratio of importance of the attraction of the poins over the rest of rules.

W_REPULSION:
(*float*) ratio of importance of the repulsion of the poins over the rest of rules.

MU:
(*float*) weight of new velocity vector over current one (used to smooth change of speed and direction).

MIN_DIST:
(*int*) minimum distance that should be between birds (i.e. birds closer than this distance are too close), in pixels.

GROUP_DIST:
(*int*) distance that determines the boundary of groups of birds (i.e. birds closer than this distance will be considered of the same group), in pixels.

VIEW_DIST:
(*int*) distance that should be between birds regarding the view rule (i.e. birds that are in the vision area of a bird and closer than this distance are too close), in pixels.

VIEW_ANGLE:
(*float*) angle that determines the vision area of a bird, in radians.

MIN_DIST_ATTRACTOR:
(*int*) distance from which attraction points will try to escape from birds (because it will "notice" them), in pixels.

MIN_DIST_REPULSOR:
(*int*) distance from which attraction points will try to go towards from birds (because it will "notice" them), in pixels.

GROUP_DIST_REPULSOR:
(*int*) distance that determines which birds are withing the group boundary of the repulsion point (so it will try to go towards the center of that group), in pixels.

WIDTH:
(*int*) width of screen, in pixels.

HEIGHT:
(*int*) height of screen, in pixels.

X_MIN:
(*int*) minimum value for x coordinate of any bird, in pixels.

X_MAX:
(*int*) maximum value for x coordinate of any bird, in pixels.

Y_MIN:
(*int*) minimum value for y coordinate of any bird, in pixels.

Y_MAX:
(*int*) maximum value for y coordinate of any bird, in pixels.

Z_MIN:
(*int*) minimum value for z coordinate of any bird, in pixels.

Z_MAX:
(*int*) maximum value for z coordinate of any bird, in pixels.

MIN_VEL:
(*int*) minimum speed of birds and points of attraction and repulsion.

MAX_VEL:
(*int*) maximum speed of birds and points of attraction and repulsion.

BOUNDARY_DELTA:
(*float*) threshold considered for the window boundary conditions.

TIME_DELTA:
(*float*) small interval of time used to update position based on velocity.

DELTA:
(*float*) a small arbitrary float.

FPS:
(*int*) determines the speed at which frames are updated.

ROTATION:
(*int*) determines the speed at which cube rotates when the keys for rotation are pressed.

CHAPTER

TWO

BIRD MODULE

Here is defined the structure of a bird in the simulation.

class bird.**Bird**(*index: int*, *position: list*, *direction: list*, *speed: float*, *type: int*)

The class that represents a bird.

Parameters

- **index** (*int*) – index that identifies bird.
- **position** (*list*) – coordinates (x,y,z) of bird.
- **direction** (*list*) – direction of bird's velocity vector with coordinates (x,y,z), as a unit vector.
- **speed** (*float*) – module of bird's velocity vector.
- **type** (*int*) – the type of object that the instance represents. Value 1 for bird, -1 for attraction point, -2 for repulsion point.

attraction(*attraction_points*)

Go towards attraction points.

Parameters **attraction_points** (*list*) – list of coordinates of the attraction points (see ATTRACTION_POINTS in *parameters*).

Returns velocity vector that responds to the attraction of the corresponding points.

Return type *list*

avoidance(*neighbours: list*)

Separate bird from neighbours that are too close.

Parameters **neighbours** (*list*) – birds that are closer to the bird than the minimum distance (see MIN_DIST in *parameters*). Birds are represented as instances of the *bird.Bird* class.

Returns velocity vector that responds to the Avoidance rule.

Return type *list*

center(*group_birds: list*)

Seek cohesion with other bird's positions. Bird will change direction to move toward the average position of all birds.

Parameters **group_birds** (*list*) – birds that are closer to the bird than the group boundary distance (see GROUP_DIST in *parameters*). Birds are represented as instances of the *bird.Bird* class.

Returns velocity vector that responds to the Center rule.

Return type *list*

copy(*group_birds: list*)

Seek cohesion with other bird's directions (average direction).

Parameters **group_birds** (*list*) – birds that are closer to the bird than the group boundary distance (see GROUP_DIST in *parameters*). Birds are represented as instances of the *bird.Bird* class.

Returns velocity vector that responds to the Copy rule.

Return type *list*

repulsion(*repulsion_points*)

Go towards repulsion points.

Parameters **repulsion_points** (*list*) – list of coordinates of the repulsion points (see REPULSION_POINTS in *parameters*).

Returns velocity vector that responds to the repulsion of the corresponding points.

Return type *list*

update(*close_neighbours*, *group_birds*, *attraction_points*, *repulsion_points*)

Updates direction, speed and position of bird, considering all rules, and the attraction and repulsion points.

Parameters

- **close_neighbours** (*list*) – birds that are closer to the bird than the minimum distance (see MIN_DIST in *parameters*). Birds are represented as instances of the *bird.Bird* class.
- **group_birds** (*list*) – birds that are closer to the bird than the group boundary distance (see GROUP_DIST in *parameters*). Birds are represented as instances of the *bird.Bird* class.
- **attraction_points** (*list*) – list of coordinates of the attraction points (see ATTRACTION_POINTS in *parameters*).
- **repulsion_points** (*list*) – list of coordinates of the repulsion points (see REPULSION_POINTS in *parameters*).

updateAttractor(*all_birds*)

Updates direction, speed and position of the attractor points. They will avoid the birds that are closer than a minimum distance (see MIN_DIST_ATTRACTOR in *parameters*).

Parameters **all_birds** (*list*) – all the birds in the simulation, represented as instances of the *bird.Bird* class.

updatePos(*diff_time*)

Update bird's position using speed and direction. Takes into consideration boundary conditions.

Parameters **diff_time** (*float*) – small interval of time used to update position based on velocity.

updateRepulsor(*all_birds*)

Updates direction, speed and position of the repulsion points. They will go towards the birds that are closer than a minimum distance (see MIN_DIST_REPULSOR in *parameters*). They will also go towards the center of the group of birds that are closer than a group boundary distance (see GROUP_DIST_REPULSOR in *parameters*).

Parameters **all_birds** (*list*) – all the birds in the simulation, represented as instances of the *bird.Bird* class.

view(*group_birds: list*)

Move if there is another bird in area of view.

Parameters **group_birds** (*list*) – birds that are closer to the bird than the group boundary distance (see GROUP_DIST in *parameters*). Birds are represented as instances of the *bird.Bird* class.

Returns velocity vector that responds to the View rule.

Return type *list*

CHAPTER

THREE

INITIALIZE_BIRDS MODULE

Initialization of birds.

initialize_birds.**generateAttractionPoints**()
Generates a list of attraction points.

Returns list of instances of the class *bird.Bird*, with the attribute `type` assigned to -1 (which represents an Attraction Point).

Return type *list*

initialize_birds.**generateBirds**()
Generates a list of birds. Positions and velocity are random.

Returns list of instances of the class *bird.Bird*.

Return type *list*

initialize_birds.**generateRepulsionPoints**()
Generates a list of repulsion points.

Returns list of instances of the class *bird.Bird*, with the attribute `type` assigned to -2 (which represents an Repulsion Point).

Return type *list*

CHAPTER

FOUR

GRAPHICS MODULE

Functions used to render graphics.

graphics.**draw_circle**(*position*, *color*, *radius=10*, *side_num=10*)

Draws a circle of a given color.

Parameters

- **position** (*list*) – coordinates of the circle's center point.
- **color** (*str*) – the color of the circle. For example: 'red', 'green'.
- **radius** (*int, optional*) – radius of the circle (in pixels), defaults to 10.
- **side_num** (*int, optional*) – radius of the circle (in pixels), defaults to 10.

graphics.**draw_cone**(*pos*, *direction*, *radius*, *height*, *slices=7*, *stacks=1*)

Draws a black cone inisde the container.

Parameters

- **pos** (*list*) – coordinates of the cone's head vertex.
- **direction** (*list*) – unitary vector that represents the direction in wich the cone is pointed.
- **radius** (*int*) – radius of the cones base, in pixels.
- **height** (*int*) – height of the cone, in pixels.
- **slices** (*int, optional*) – Number of slices that will determine the cone's shape in the graphics, defaults to 7.
- **stacks** (*int, optional*) – Number of stacks that will determine the cone's shape in the graphics, defaults to 1.

graphics.**draw_container**()

Draws 2D square or 3D cube that contains birds.

graphics.**draw_sphere**(*position*, *color*, *r=10*, *lats=10*, *longs=10*)

Draws a sphere of a given color.

Parameters

- **position** (*list*) – coordinates of the sphere's center point.
- **color** (*str*) – the color of the sphere. For example: 'red', 'green'
- **r** (*int, optional*) – radius of the sphere (in pixels), defaults to 10.
- **lats** (*int, optional*) – number of lats that will determine the sphere's shape in the graphics, defaults to 10.
- **longs** (*int, optional*) – number of longs that will determine the sphere's shape in the graphics, defaults to 10.

graphics.**draw_triangle**(*head*, *tail_vertex1*, *tail_vertex2*)

Draws a black triangle inisde the container.

Parameters

- **head** (*list*) – coordinates of the triangle's head vertex.
- **tail_vertex1** (*list*) – coordinates of a tail's vertex.
- **tail_vertex2** (*list*) – coordinates of the other tail's vertex.

graphics.**initialize_window**()

Initializes window where simulation will be shown when running the program.

CHAPTER

FIVE

MAIN MODULE

File where simulation is runned.

main.**main**()
Function that has to be executed to run the simulation.