

Using Game Maker to Simulate Interacting Wolf-Bunny Populations

Mike Bailey

mjb@cs.oregonstate.edu

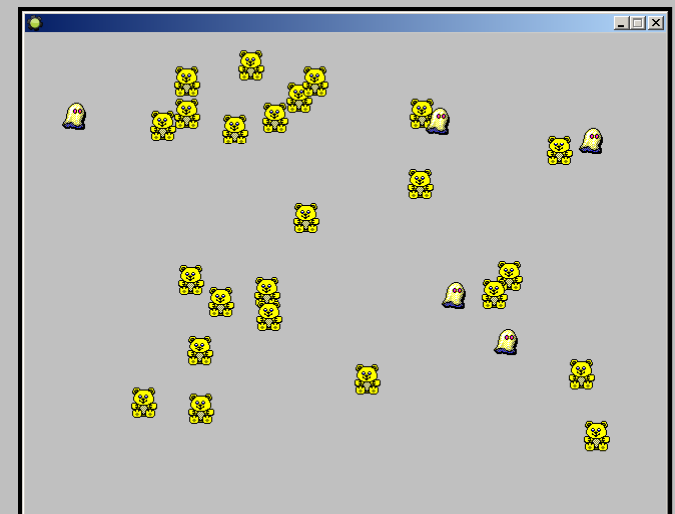
Oregon State University



Wolf



Bunny



Fundamentals of Population Growth

Each population has a *Carrying Capacity*, K , which says what that population's growth limit will be.

The Carrying Capacity is a function of a lot of things: the population size, the food source size, other populations for which this one is their food source, the weather, disease, invasive species, global warming, human habitat encroachment, etc. Some researchers spend their entire career trying to figure out what the Carrying Capacity of a certain population is.

In this simulation, we will assume that:

1. The Carrying Capacity for bunnies reflects a maximum value when their vegetation food supply can no longer support the population.
2. The Carrying Capacity for bunnies goes down when the wolf population goes up, and vice versa.
3. The Carrying Capacity for wolves goes down when the bunny population goes down, and vice versa.

More on this later.



Oregon State University
Computer Graphics



Wolf

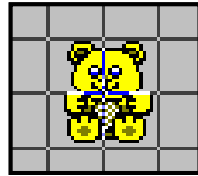


Bunny

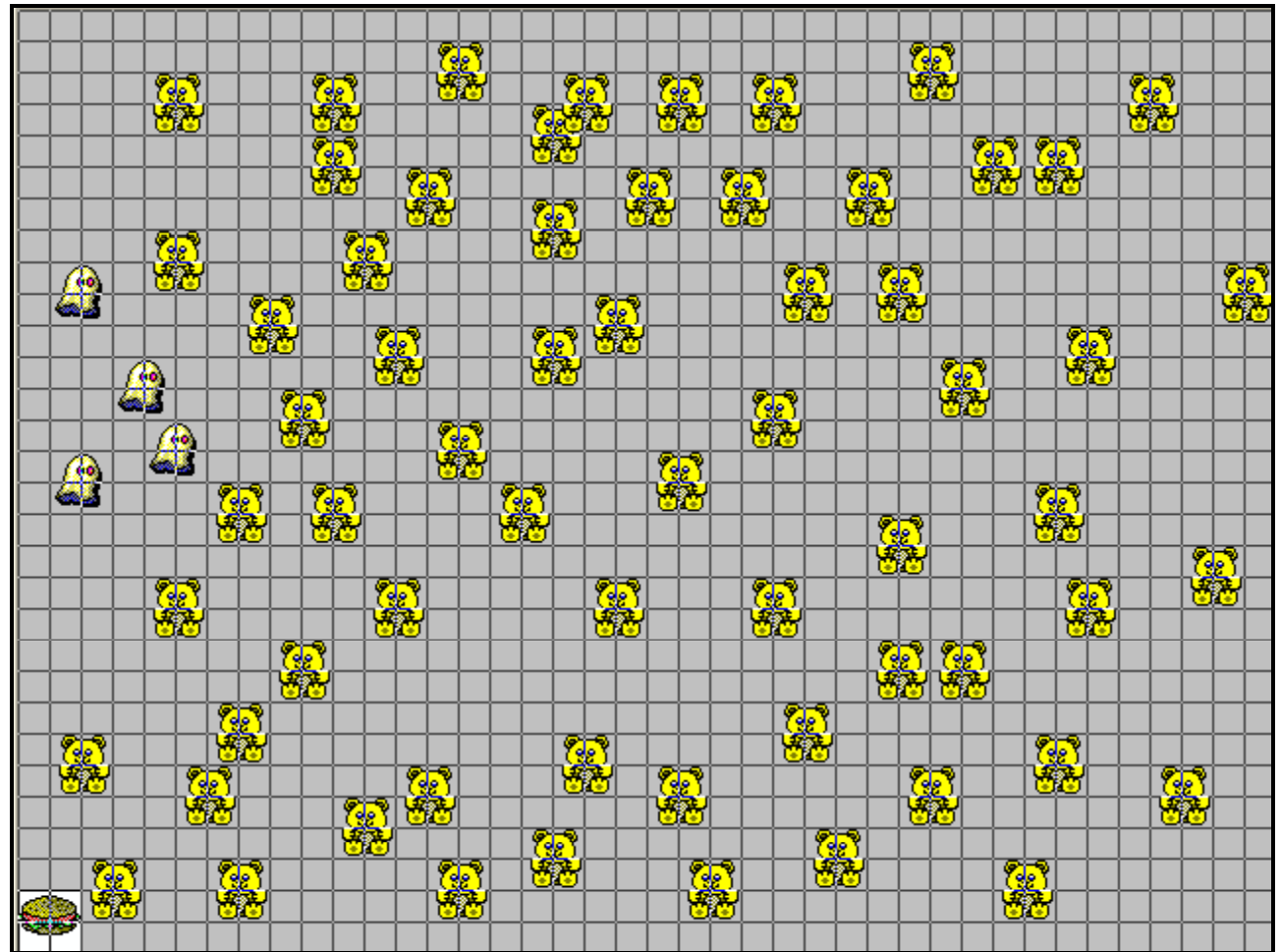
The Forest to Start



Wolf

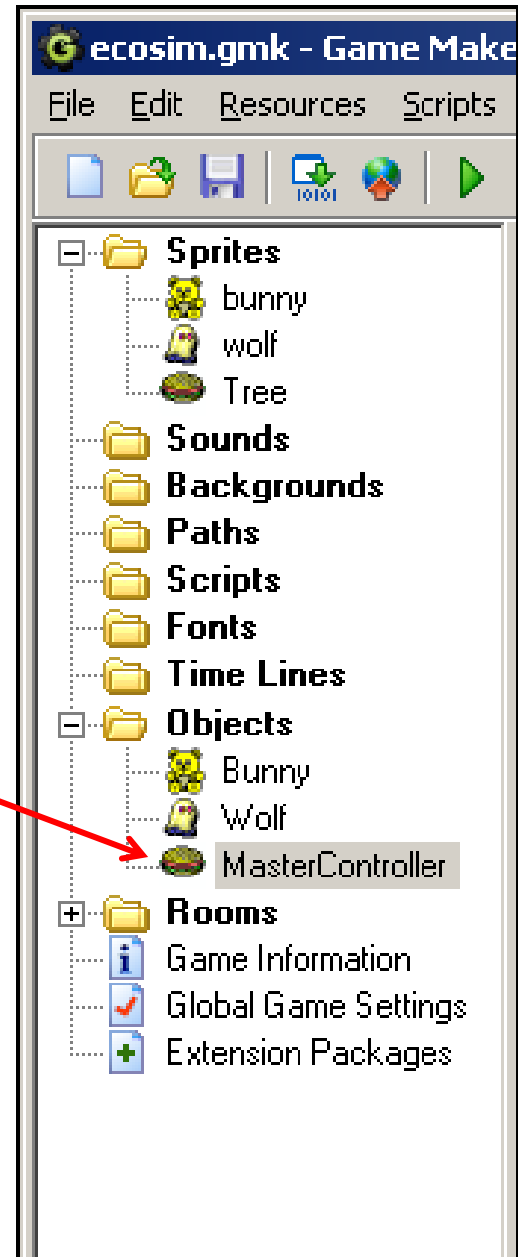


Bunny

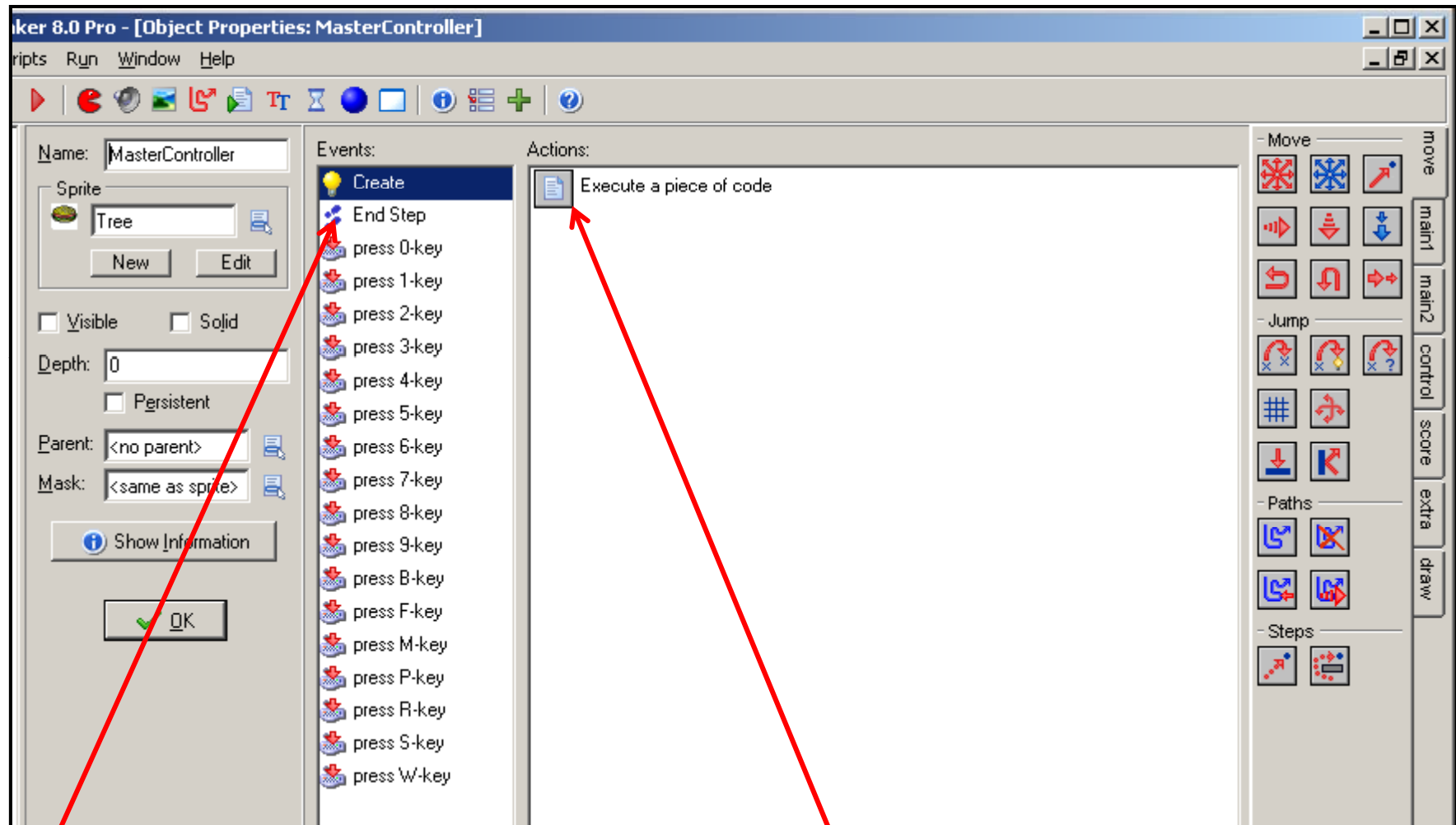


Let's Look at How the Simulation Works

Double-
click here



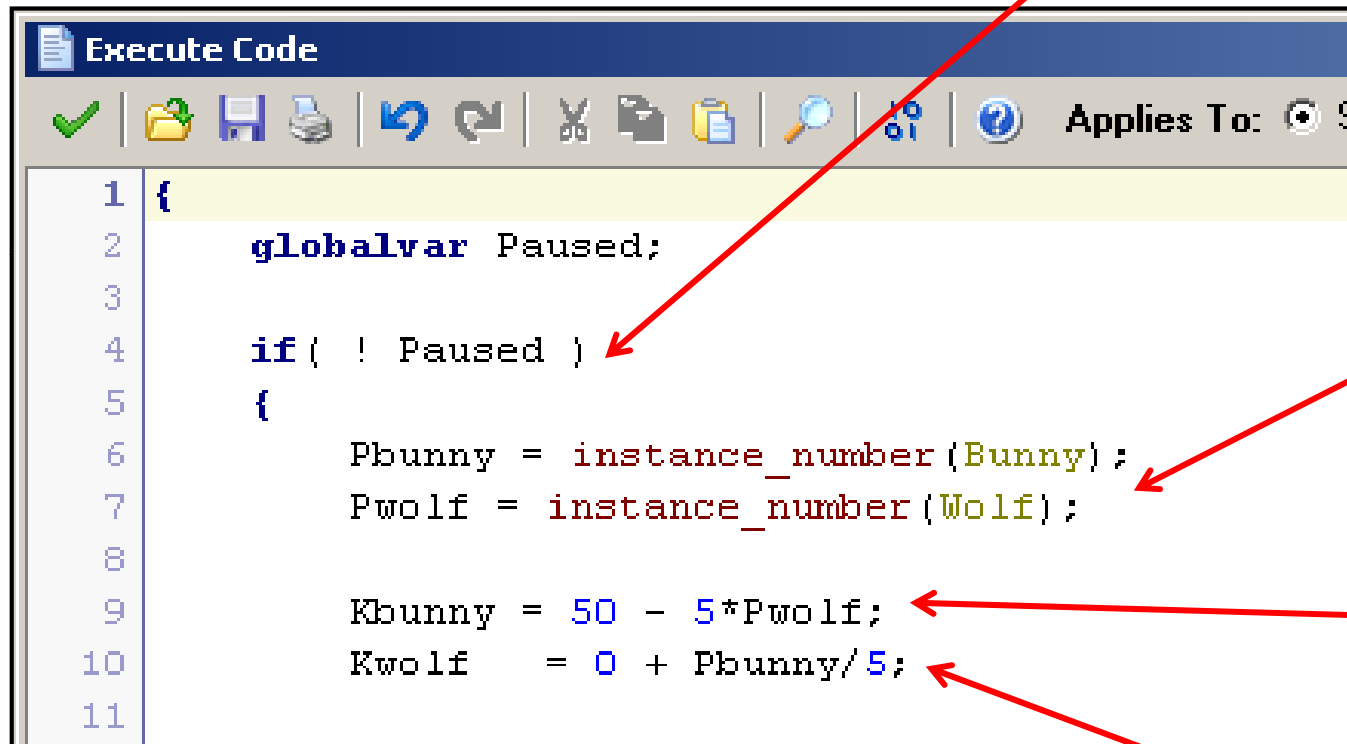
Let's Look at How the Simulation Works



Then, single-click here ...

... and double-click here

Let's Look at How the Simulation Works



```
1 {
2     globalvar Paused;
3
4     if( ! Paused )
5     {
6         Pbunny = instance_number(Bunny);
7         Pwolf = instance_number(Wolf);
8
9         Kbunny = 50 - 5*Pwolf;
10        Kwolf   = 0 + Pbunny/5;
11    }
```

If the simulation is not currently paused
(the exclamation point is computer-
programming for “not”).

Find out how many
bunnies and wolves there
are now

Compute the bunnies’
current Carrying
Capacity.

Compute the wolves’
current Carrying
Capacity.

Let's Look at How the Simulation Works

If the bunnies' population is less than their current Carrying Capacity, reproduce a new one. (But only do this if there are more than zero bunnies around to do it.)

```
11
12     if( Pbunny < Kbunny  &&  Pbunny > 0 )
13     {
14         instance_create( random_range(20, room_width-20), random_range( 20, room_height-20), Bunny );
15     }
16
17     if( Pbunny > Kbunny )
18     {
19         if( Pbunny > 0 )
20         {
21             with( instance_find( Bunny, 0 ) )
22                 instance_destroy();
23         }
24     }
25
26     if( Pbunny == Kbunny )
27     {
28         instance_create( random_range(20, room_width-20), random_range( 20, room_height-20), Bunny );
29         with( instance_find( Bunny, 0 ) )
30             instance_destroy( );
31     }
32
```

If the bunnies' population exceeds their current Carrying Capacity, delete one.

If the bunnies' population is the same as their current Carrying Capacity, then leave the population number alone. But, in this case, we will both create and destroy one. Otherwise, it will look like the simulation is frozen.

Let's Look at How the Simulation Works

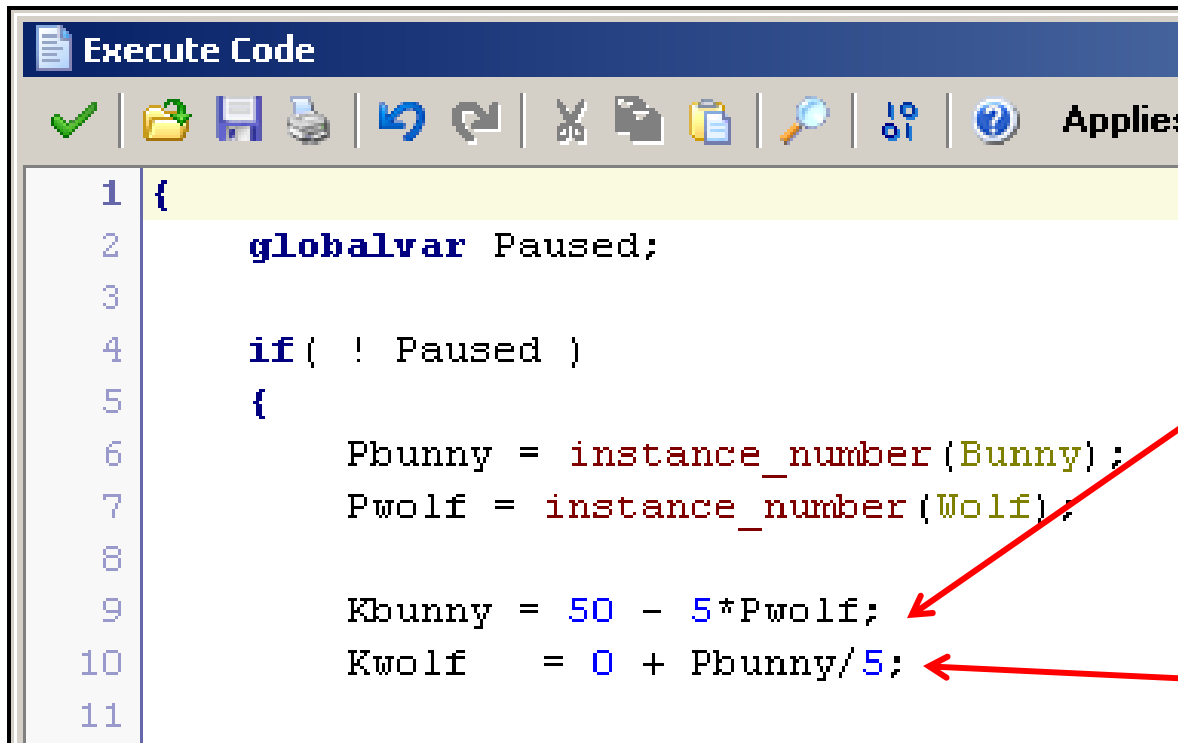
If the wolves' population is less than their current Carrying Capacity, reproduce a new one. (But only do this if there are more than zero wolves around to do it.)

```
33
34     if( Pwolf < Kwolf  &&  Pwolf > 0 )
35     {
36         instance_create(  random_range(20, room_width-20), random_range( 20, room_height-20), Wolf );
37     }
38
39     if( Pwolf > Kwolf )
40     {
41         if( Pwolf > 0 )
42         {
43             with( instance_find( Wolf, 0 ) )
44                 instance_destroy( );
45         }
46     }
47
48     if( Pwolf == Kwolf )
49     {
50         instance_create(  random_range(20, room_width-20), random_range( 20, room_height-20), Wolf );
51         with( instance_find( Wolf, 0 ) )
52             instance_destroy();
53     }
54 }
55
56 }
```

If the wolves' population exceeds their current Carrying Capacity, delete one.

If the wolves' population is the same as their current Carrying Capacity, then leave the population number alone. But, in this case, we will both create and destroy one. Otherwise, it will look like the simulation is frozen.

The Really Interesting Part is Computing the *Carrying Capacities*



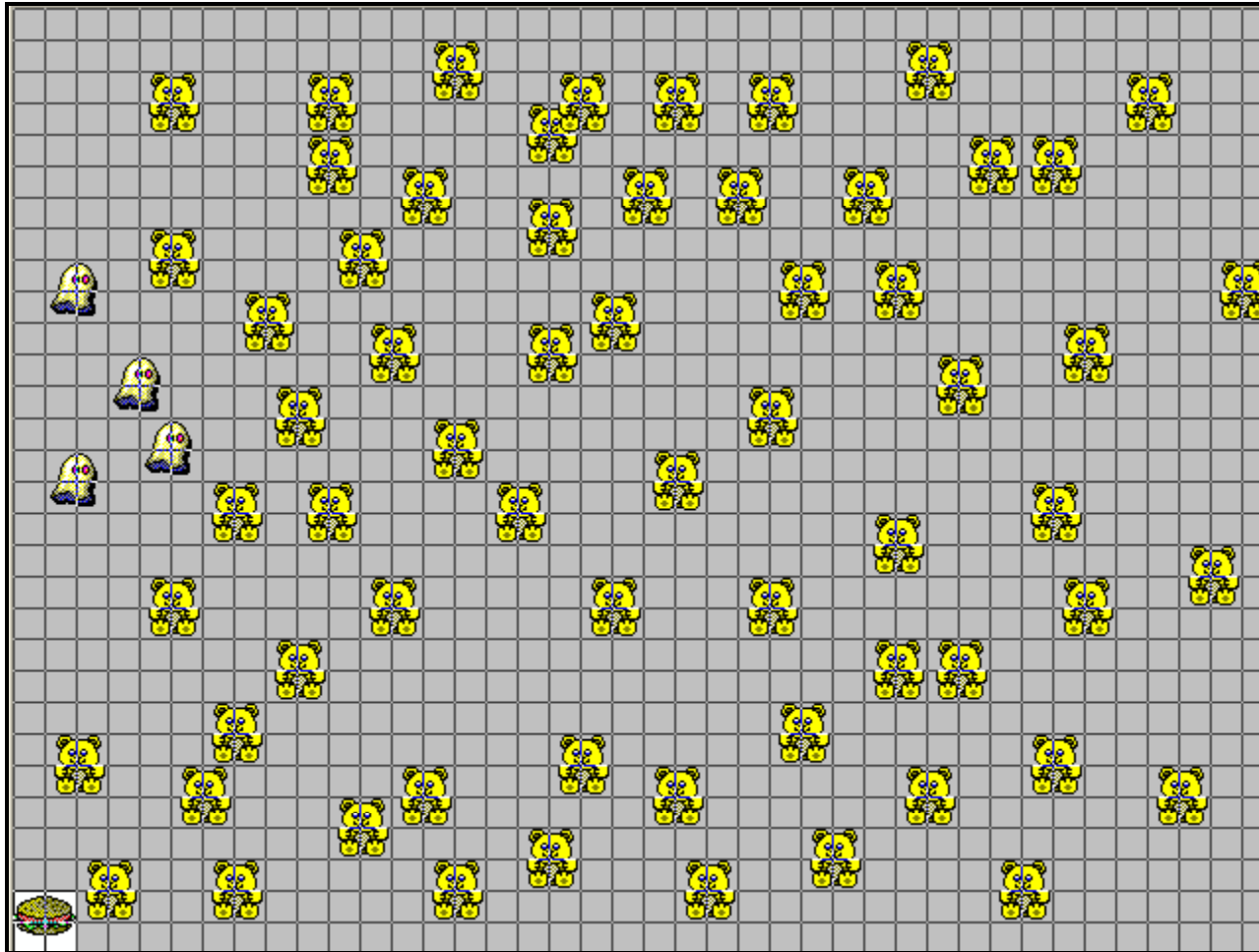
```
1 {
2   globalvar Paused;
3
4   if( ! Paused )
5   {
6     Pbunny = instance_number(Bunny);
7     Pwolf = instance_number(Wolf);
8
9     Kbunny = 50 - 5*Pwolf;
10    Kwolf = 0 + Pbunny/5;
11  }
```

Let's assume the carrying capacity of the bunnies is 50 minus 5 times the number of wolves. If this is greater than the current bunny population, the bunny population will go up. If less, it will go down

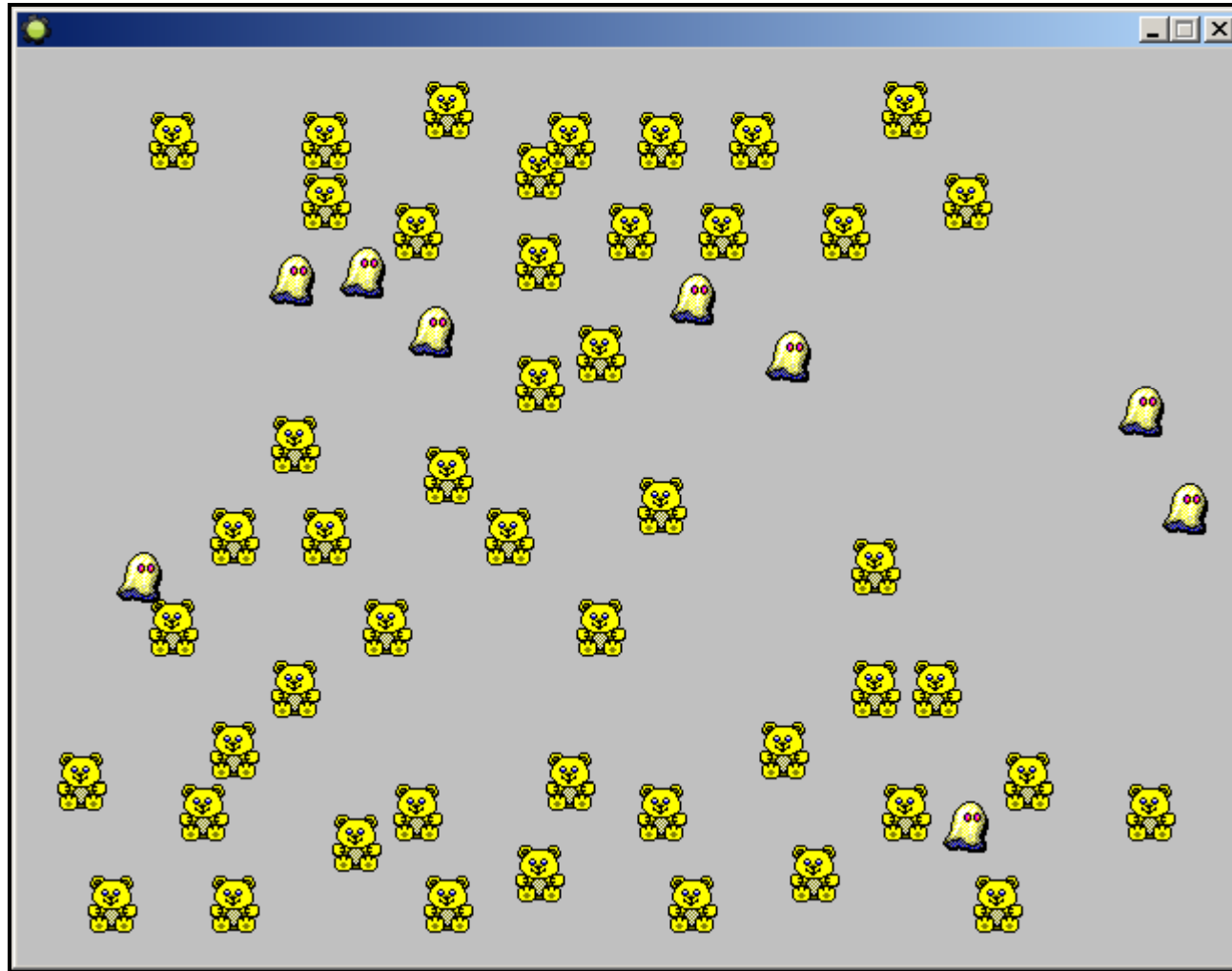
Let's assume the carrying capacity of the wolves is 0 plus the number of bunnies divided by 5. If this is greater than the current wolf population, the wolf population will go up. If less, it will go down.

These assumptions are arbitrary on our part, but they will give us reasonable results. Feel free to play around with this and see what kind of different results you can get.

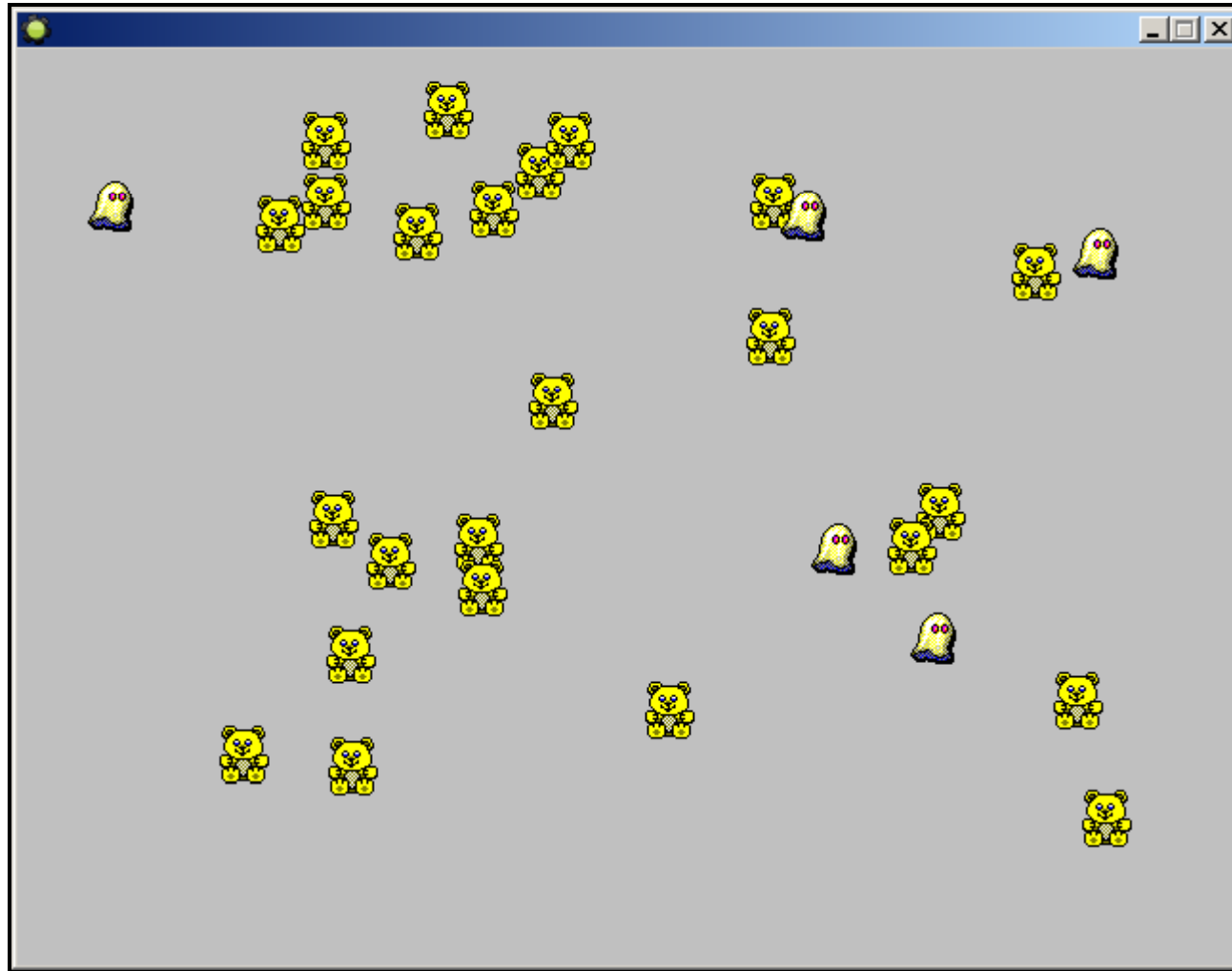
Let's Try It – Here is the Forest to Start



The Forest after a Few Steps



The Forest Hits a Loose Equilibrium

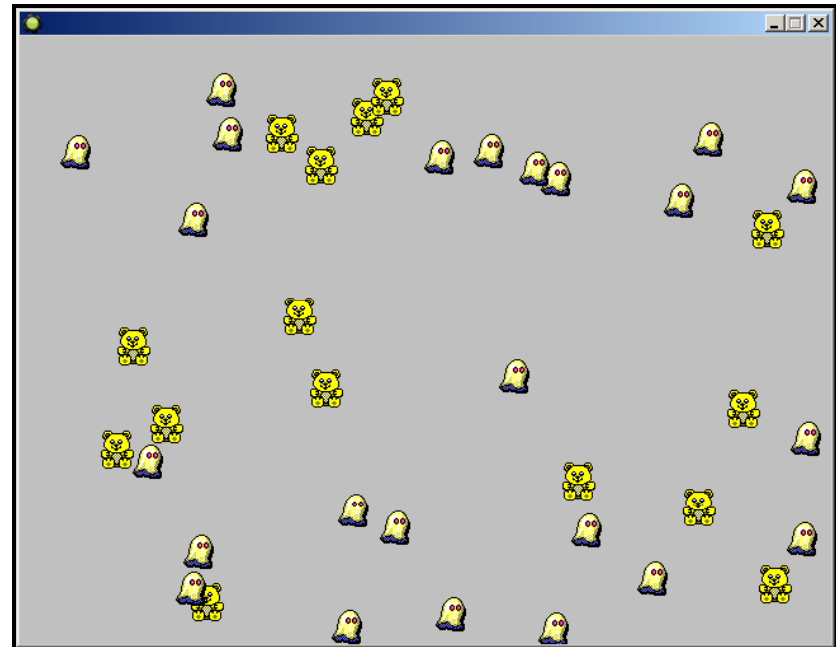
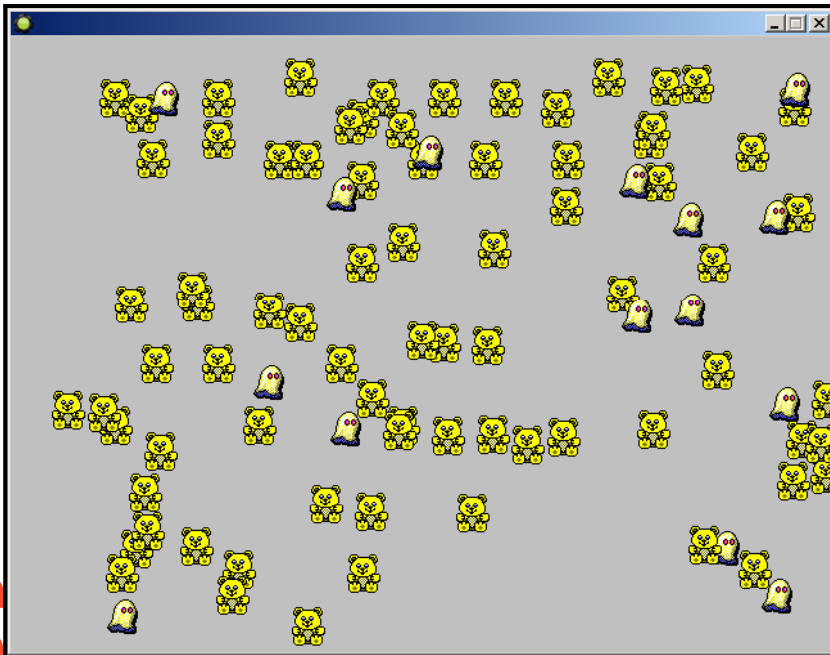


Try adding or deleting some wolves and bunnies from the forest while the simulation is running. What do you think will happen? Why?

To add bunnies while the simulation is running, type a number and the letter **b**. For example, to add 10 bunnies, you would type: **1-0-b** (without the dashes)

To subtract 10 bunnies, indicate that you are adding a minus number and type: **m-1-0-b**

You can do the same with the wolves instead by using the letter **w**. For example: **1-0-w** or **m-1-0-w**



Keyboard Commands You Might Find Useful

- 0-9** Use the digits to enter a number of bunnies or wolves to add or delete.
- f** Make the simulation run faster. This is useful to “cut to the chase” and see what the effect of a certain situation will be.
- m** Make the number you are entering minus (i.e., negative). This is so you can eliminate some bunnies or wolves.
- p** Pause the simulation. This is useful if you want to count the number of wolves and bunnies. Hit the **p** key again to continue;
- r** Reset the simulation to the way things were at the start
- s** Make the simulation run slower. This is useful to slow the time steps down so you can better see what is happening at each step.

Given these Carrying Capacities, What Will the Equilibrium Look Like?

```
6      Pbunny = instance_number(Bunny);  
7      Pwolf  = instance_number(Wolf);  
8  
9      Kbunny = 50 - 5*Pwolf;  
10     Kwolf  = 0 + Pbunny/5;  
11
```

This is easy to compute. At equilibrium, each population equals its Carrying Capacity.

So, from there, we know that at equilibrium:

$$B = 50 - 5W \quad (1)$$

And, from there, we know that at equilibrium:

$$W = B / 5 \quad (2)$$

Substituting (2) into (1):

$$B = 50 - B$$

Solving and substituting gives $B^* = 25$ and $W^* = 5$, which you see (approximately) if you pause the simulation after a while and count the populations.