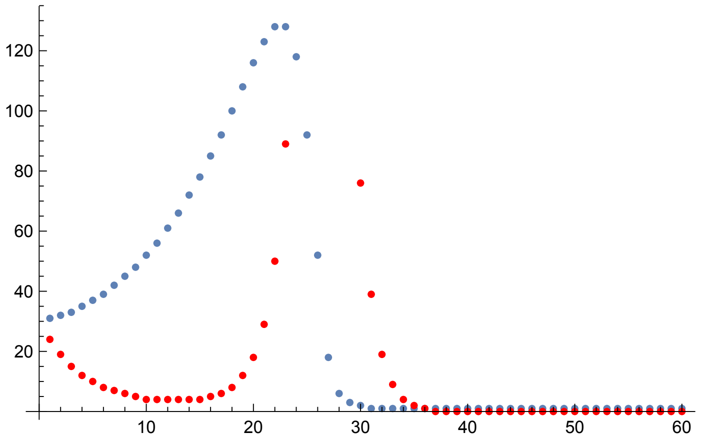
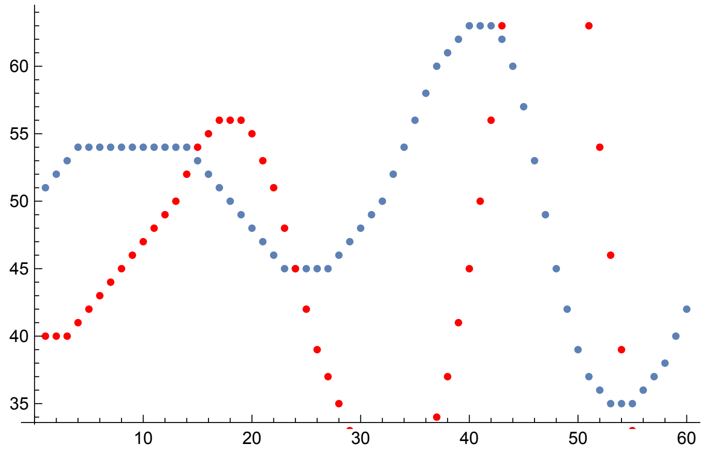
This is the Kitten/Puppy growth population being used in the game. It is two difference equations based on the Lotka-Volterra predator-prey mathematical model. It is an iterative process that predicts the next generation of cats and dogs based on the previous generation.

```
In[766]:= catAndDogGrowth[startingCat_, startingDog_, numberOfDays_] :=
           Module[{cat = startingCat, dog = startingDog,
             catVelocity = 0, dogVelocity = 0}, catAndDog = Table[{
               dogNew = 0.5 * dog + 0.01 * cat * dog;
               catNew = 1.1 * cat - 0.002 * cat * dog;
               cat = If[catNew < 0, 0, IntegerPart[catNew]],
               dog = If[dogNew < 0, 0, IntegerPart[dogNew]], i}, {i, 1, numberOfDays}];
            Show[{ListPlot[Transpose[{catAndDog[[All, 3]], catAndDog[[All, 1]]}]],
              ListPlot[Transpose[{catAndDog[[All, 3]], catAndDog[[All, 2]]}],
               PlotStyle → RGBColor[1, 0, 0]]}]]

In[779]:= catAndDogGrowth[30, 30, 60]

Out[779]=
```



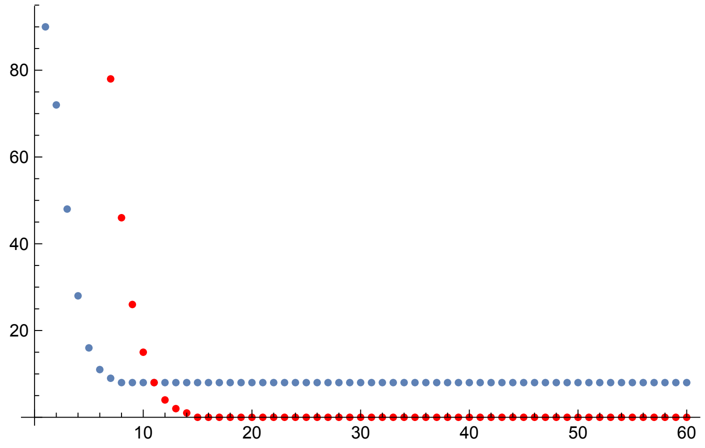
In[770]:= `catAndDogGrowth[50, 40, 60]`

Out[770]=



In[769]:= `catAndDogGrowth[100, 100, 60]`

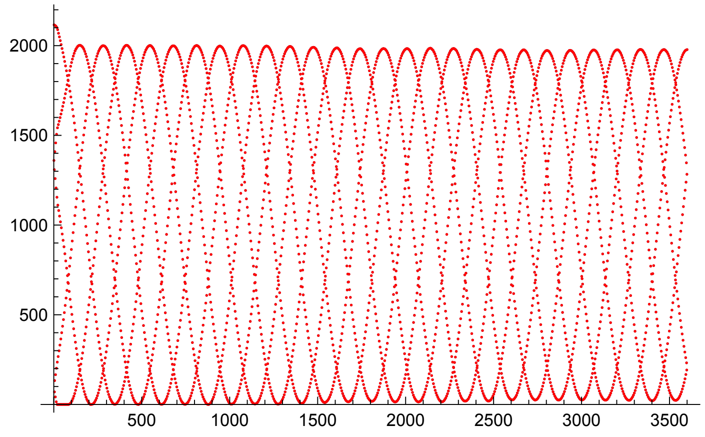
Out[769]=



This is a more abstract model that I made up while doing the game jam, but I'm sure it exists elsewhere. The idea is that the dog population is always tending towards an ideal dog population, and the cat population is always tending towards an ideal dog population, but the two interfere with one another. Both dogs and cats tend towards their ideal population, but overshoot and readjust by Hooke's law of F = -kx this is why we get sinusoidal graphs.

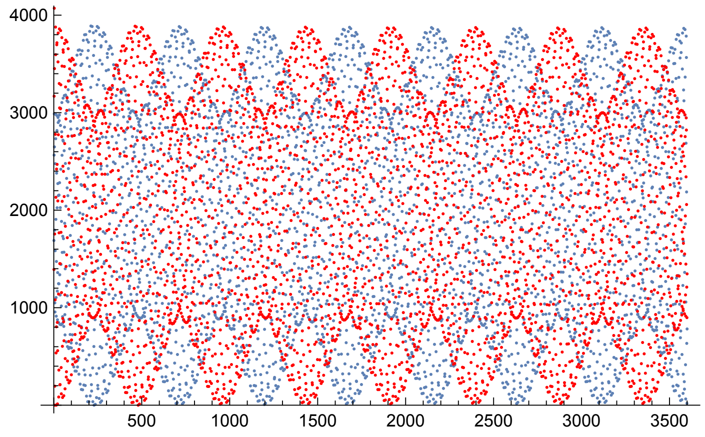
```
In[687]:= catAndDogGrowth2[startingCat_, startingDog_, numberOfDays_, k_, a_, d_] :=
  Module[{cat = startingCat, dog = startingDog,
    catVelocity = 0, dogVelocity = 0}, catAndDog = Table[{
      catVelocity = (catVelocity + (k * (2000 - cat)) - a * (cat) - d * dog);
      catNew = cat + catVelocity;
      dogVelocity = (dogVelocity + (k * (2000 - dog)) - a * (cat) - d * dog);
      dogNew = dog + (dogVelocity);
      cat = If[catNew < 0, 0, IntegerPart[catNew]],
      dog = If[dogNew < 0, 0, IntegerPart[dogNew]], i}, {i, 1, numberOfDays}];
   Show[{ListPlot[Transpose[{catAndDog[[All, 3]], catAndDog[[All, 1]]}]],
     ListPlot[Transpose[{catAndDog[[All, 3]], catAndDog[[All, 2]]}],
      PlotStyle → RGBColor[1, 0, 0]]}]]

In[688]:= catAndDogGrowth2[100, 100, 3600, .7, .2, .5]
```

Out[688]=



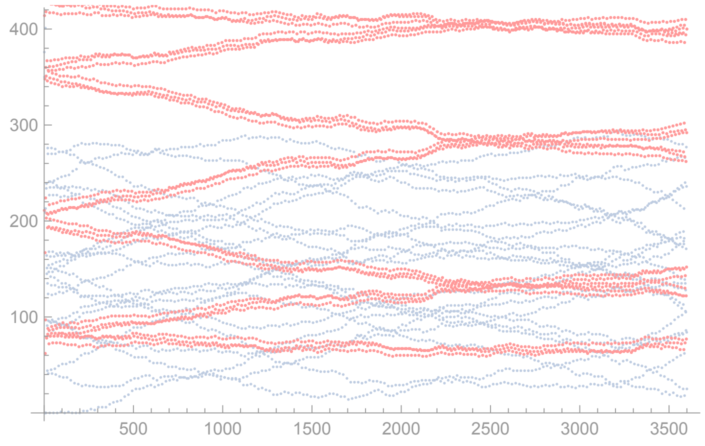
```
In[713]:= catAndDogGrowth2[1000, 10, 3600, .7, .01, .01]
```

Out[713]=



This algorithm is similar to the last, but more modular. I’m really just trying to make neat looking graphs at this point...

```
In[714]:= catAndDogGrowth3[startingCat_, startingDog_,
            numberOfDays_, catConstant_, dogConstant_, a_, b_, c_, d_] :=
          Module[{cat = startingCat, dog = startingDog, catVelocity = 0, dogVelocity = 0},
            catAndDog = Table[{
                catVelocity = (catVelocity + (catConstant * (300 - cat)) - a * (cat) - b * dog);
                catNew = cat + catVelocity;
                dogVelocity = (dogVelocity + (dogConstant * (300 - dog)) - c * (cat) - d * dog);
                dogNew = dog + (dogVelocity);
                cat = If[catNew < 0, 0, IntegerPart[catNew]],
                dog = If[dogNew < 0, 0, IntegerPart[dogNew]], i}, {i, 1, numberOfDays}];
            Show[{ListPlot[Transpose[{catAndDog[[All, 3]], catAndDog[[All, 1]]}]],
              ListPlot[Transpose[{catAndDog[[All, 3]], catAndDog[[All, 2]]}],
                PlotStyle → RGBColor[1, 0, 0]]}]]

          catAndDogGrowth3[10, 11, 3600, 1.3, .7, .50, .50, .1, .1]

Out[747]=
```



```
In[746]:= catAndDogGrowth3[100, 10, 3600, 1.1, .7, .50, .50, .1, .11]

Out[746]=
```