```
hh[{x_, y_}] = {y, y^2 + c - a x} /. a → .15 /. c → -1.1;
hh2[{x_, y_}] = hh[hh[{x, y}]];
dhh = Transpose[{∂x hh[{x, y}], ∂y hh[{x, y}]}];
dhh2 = Transpose[{∂x hh[hh[{x, y}]], ∂y hh[hh[{x, y}]]}];
fixpoints = NSolve[hh[{x, y}] == {x, y}, {x, y}];
period2cycle = NSolve[hh[hh[{x, y}]] == {x, y}, {x, y}];
center = {x, y} /. fixpoints[[1]];
centerOther = {x, y} /. fixpoints[[2]];
pushvector = Eigensystem[dhh /. fixpoints[[1]]][[2, 1]];
pushvectorOther = Eigensystem[dhh /. fixpoints[[2]]][[2, 1]];
sink = {x, y} /. period2cycle[[1]];
centerOther = {x, y} /. fixpoints[[2]];
pushvector2 = Eigensystem[dhh /. fixpoints[[2]]][[2, 1]];
```

```
Eigensystem[dhh /. fixpoints[[1]]]
```

```
{{3.49931, 0.0428656}, {{-0.274771, -0.96151}, {-0.999083, -0.0428263}}}
```

```
Eigensystem[dhh /. fixpoints[[2]]]
```

```
{{-1.10663, -0.135547}, {{-0.670458, 0.741947}, {0.990938, -0.134319}}}
```

We define the Henon map "hh" with parameters "a" and "c".
We need to check the fixpoints separately to see which one is the sink and which one is the saddle. The pushvector is chosen to be the expanding eigenvalue at the saddle point. The stopping criterion is either getting large or getting close to the sink.

We color according to the argument of y. We divide by a large power of 2 in order to make the argument move less rapidly. We do not need to worry about the size of | y | because somehow that is "implicit" in the argument changes of y.

To make additional pictures, you can solve for fixed points of higher period. Then you check for the expanding eigenvector of the differential (of the iterate).

```
maxIter = 63;
maxXXSize = 600;
maxYYSize = 600;

smalldelta = .05 / maxXXSize;

newfun[{{x_, y_}, n_}] = {hh[{x, y}], n + 1};

eNorm[{x_, y_}] = Sqrt[Abs[x^2] + Abs[y^2]];
newtest[{{x_, y_}, n_}] = (Abs[y] < 10 000) && (eNorm[{x, y} - sink] > .001);

newcolorassign[{{x_, y_}, n_}] = Which[
   Abs[y] < 3, {0, 0, 0},
   Mod[Floor[Arg[y] / 2^16], 2] > 0, {.5, .5, .5} ,
    True, {1, 1, 1}];

Graphics[
 {Raster[Table[newcolorassign[NestWhile[newfun, {center - pushvector * smalldelta * I *
          (k - maxXXSize / 2 + I * (j - maxYYSize / 2))), 0}, newtest, 1, maxIter]],
    {k, 1, maxXXSize} , {j, 1, maxYYSize}]], PointSize[12 / maxXXSize],
  White, Point[{(maxXXSize - 1) / 2, (maxYYSize - 1) / 2}],
  PointSize[4 / maxXXSize], Black,
  Point[{(maxXXSize - 1) / 2, (maxYYSize - 1) / 2}]} ]
```