Prediction options

p0 – no prediction

Program computes iterations for each pixel. The output is absolutely accurate, but it is slow.

Animation time: **1**min **7**secs

Figure 1: p0 - no prediction

p1 – simple prediction

Default option

Program loads 3 pixels in row, computes first and third. If the number of iterations is the same, it predicts that the second pixel will be the same as first and third one. Output is very accurate.

In some cases, this can be slower than **p0**.

Animation time: **49**secs

Figure 2: p1 - simple prediction

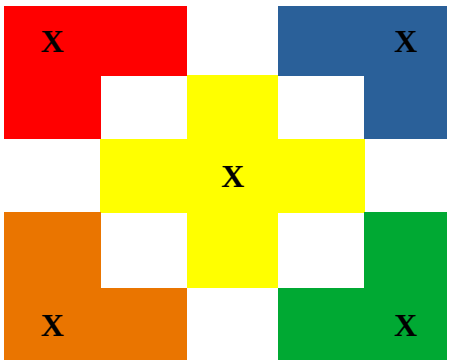
p2 – fast prediction

Always computes only 20% of pixels. Loads 25 pixels (5x5), computes 5 of them and predicts the rest.

Output is not so accurate.

Animation time: **21**secs

Figure 3: p2 - fast prediction



p3 – very-fast prediction

Always computes only 8% of pixels. Loads 100 pixels (10x10), computes 8 of them and predicts the rest.

Animation time: **11**secs

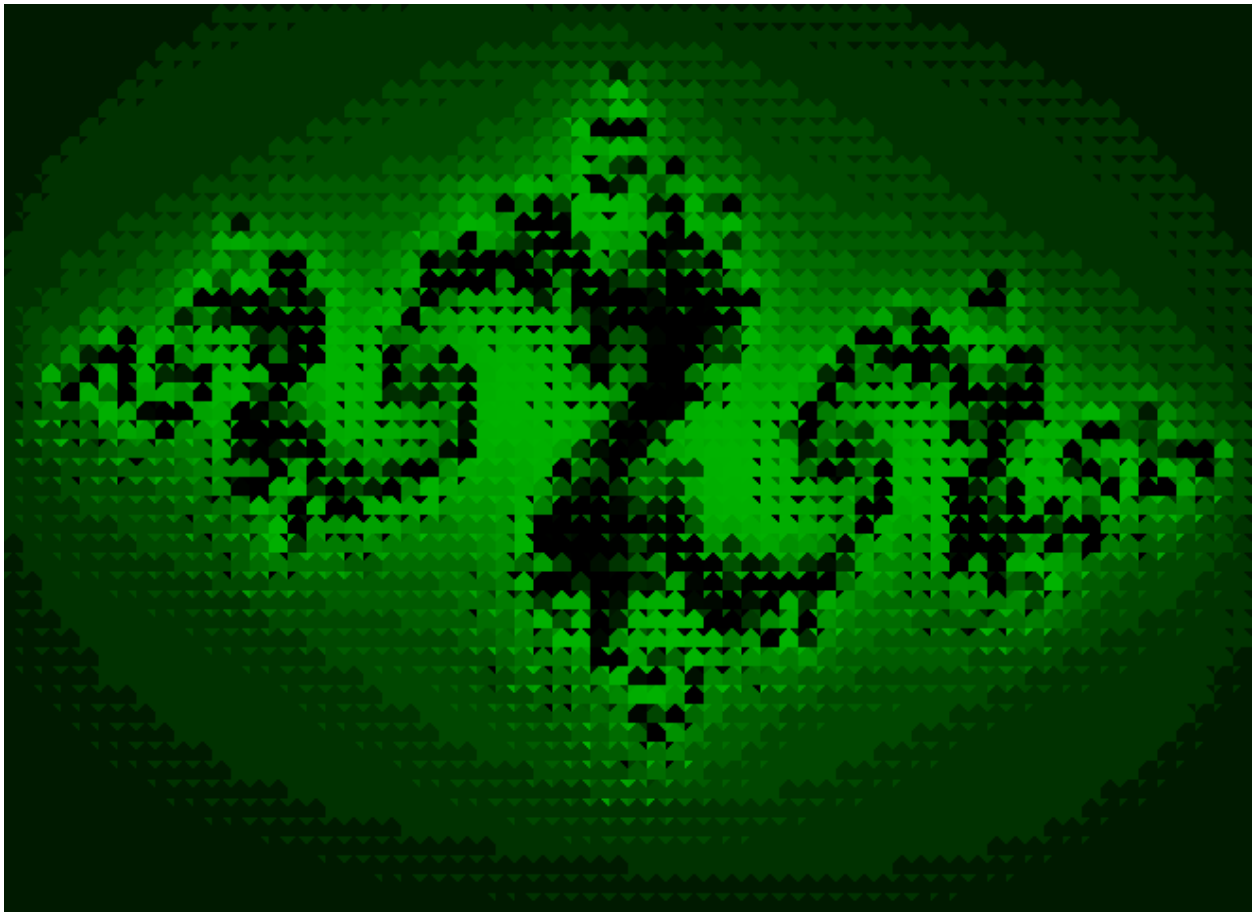
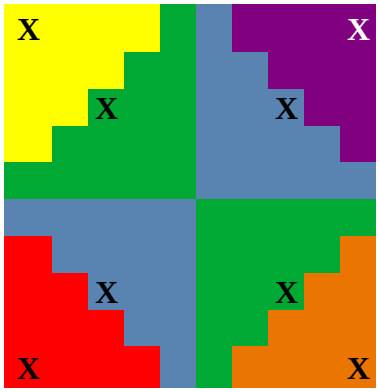


Figure 4: p3 - very-fast prediction



p4 – super-fast prediction

Always computes only 4% of pixels. Loads 100 pixels (10x10), computes 4 of them and predicts the rest.

Animation time: **9**secs

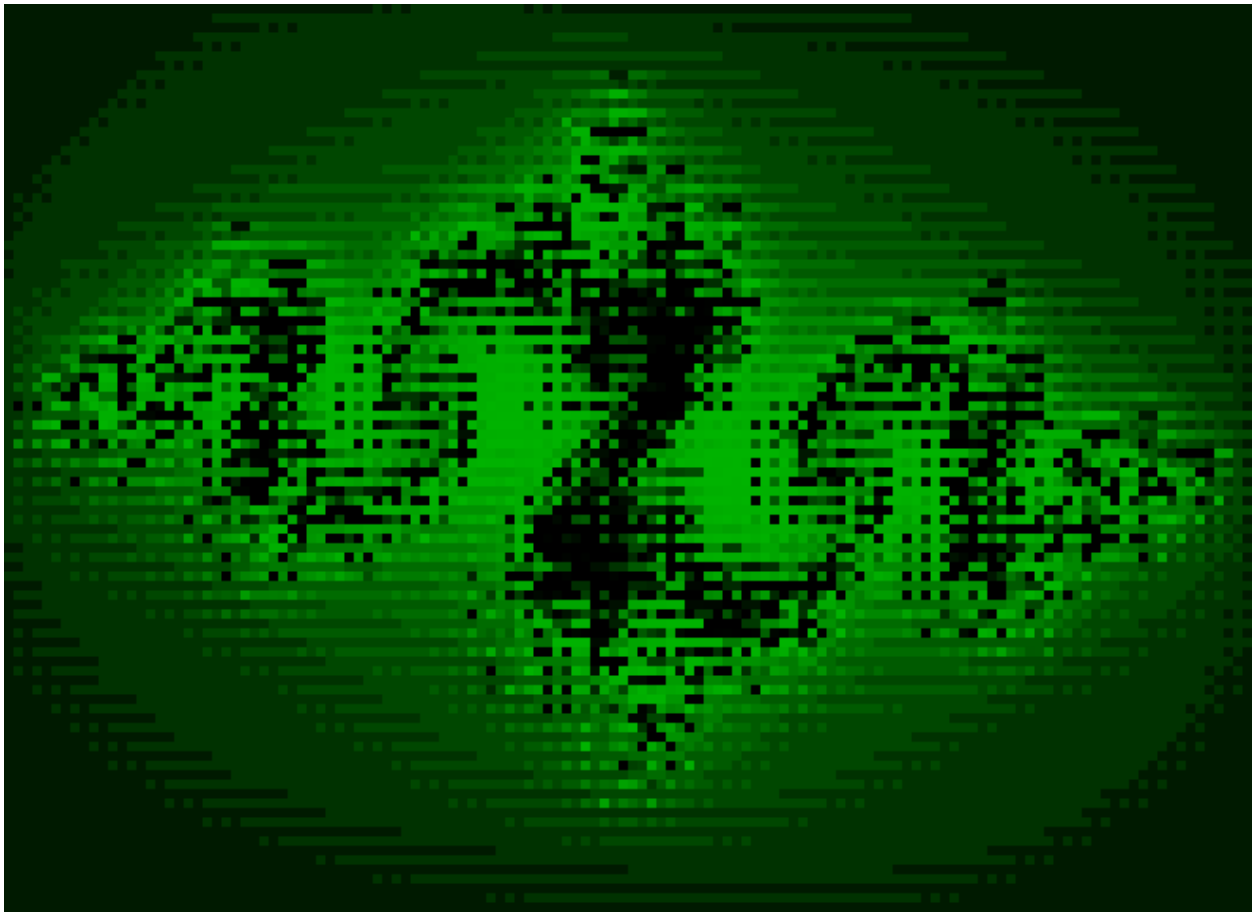
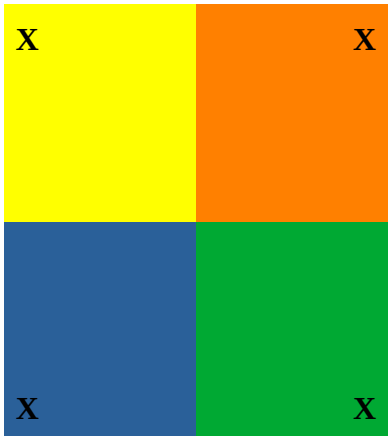


Figure 5: p4 - super-fast prediction



p5 – ultra-fast prediction

Always computes only 2% of pixels. Loads 100 pixels (10x10), computes 2 of them and predicts the rest.

Animation time: **8**secs

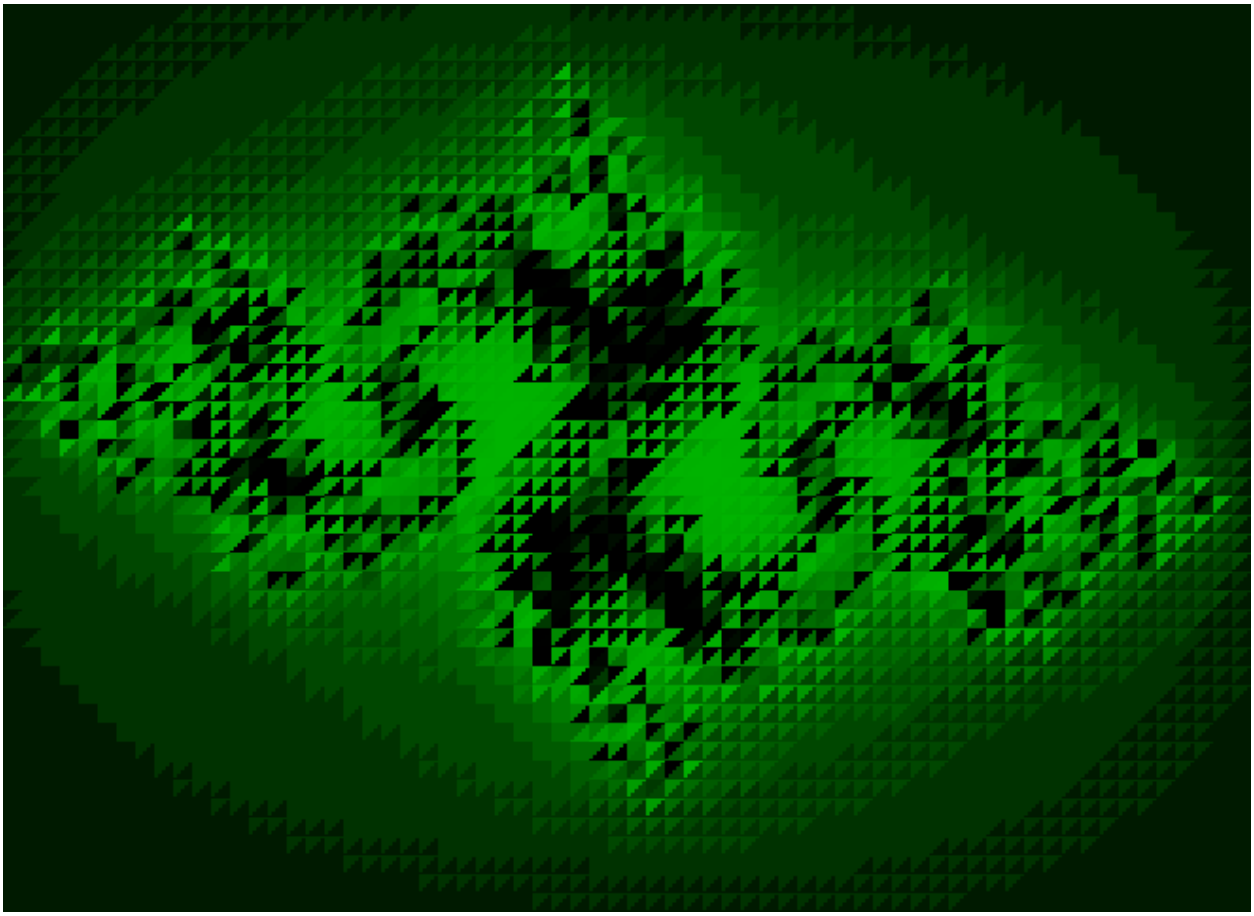
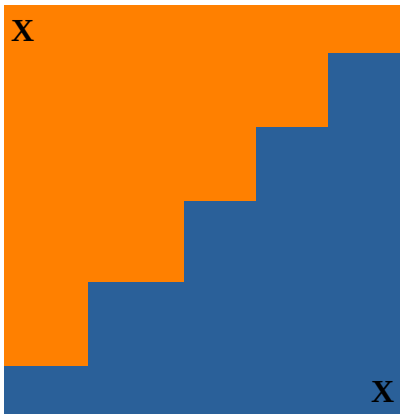


Figure 6: p5 - ultra-fast prediction



p10 – animation prediction

Computes chosen part of pixels. While computing animation, computes only

$$100 \cdot \frac{1}{distance\,between\,pixels}\%$$

of each frame. Rest remains from previous frames.

When the frames are small (fps rate is not bad) and *distance_between_pixels* is not very high, animation looks accurate.

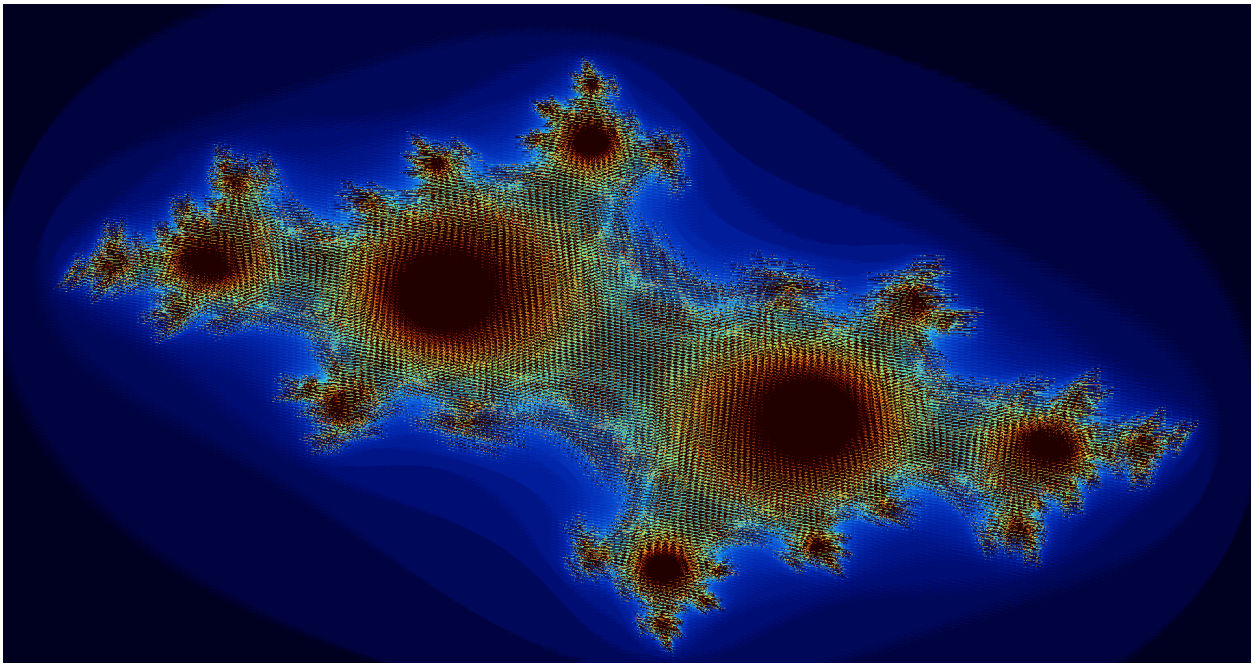


Figure 7: p10 - animation prediction