Факултет по математика и информатика
Пловдивски университет „Паисий Хилендарски“

НЕФЕРТИТИ В СЪВРЕМЕННИЯ СВЯТ

СИМБИОЗА МЕЖДУ КУЛТУРАТА НА ДРЕВЕН ЕГИПЕТ И ПОП АРТ СТИЛА

Теория на генеративното изкуство

Автор: Никол Манолова

СЪДЪРЖАНИЕ

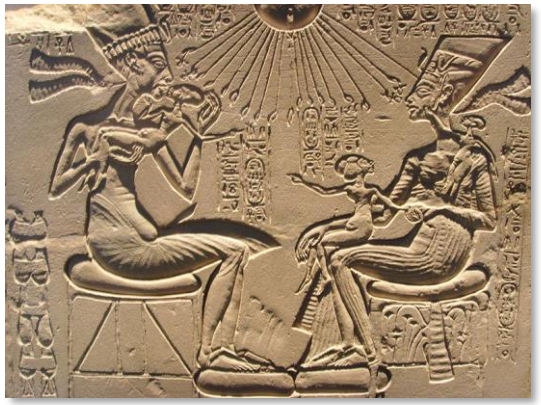
ВЪВЕДЕНИЕ

ДРЕВЕН ЕГИПЕТ

ИДЕЯ И ИСТОРИЧЕСКИ ПРЕПРАТКИ

ПОП АРТ СТИЛ

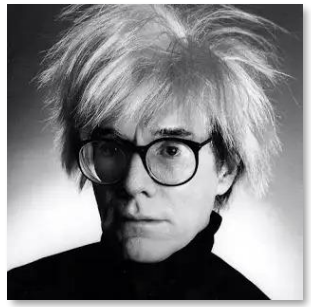
Аменхотеп IV (Ехнатон) и Нефертити



Управление: 1353 – 1336 г. пр.н.е. [1]

ПРОЕКТ

Анди Уорхол (1928 – 1987 г.)

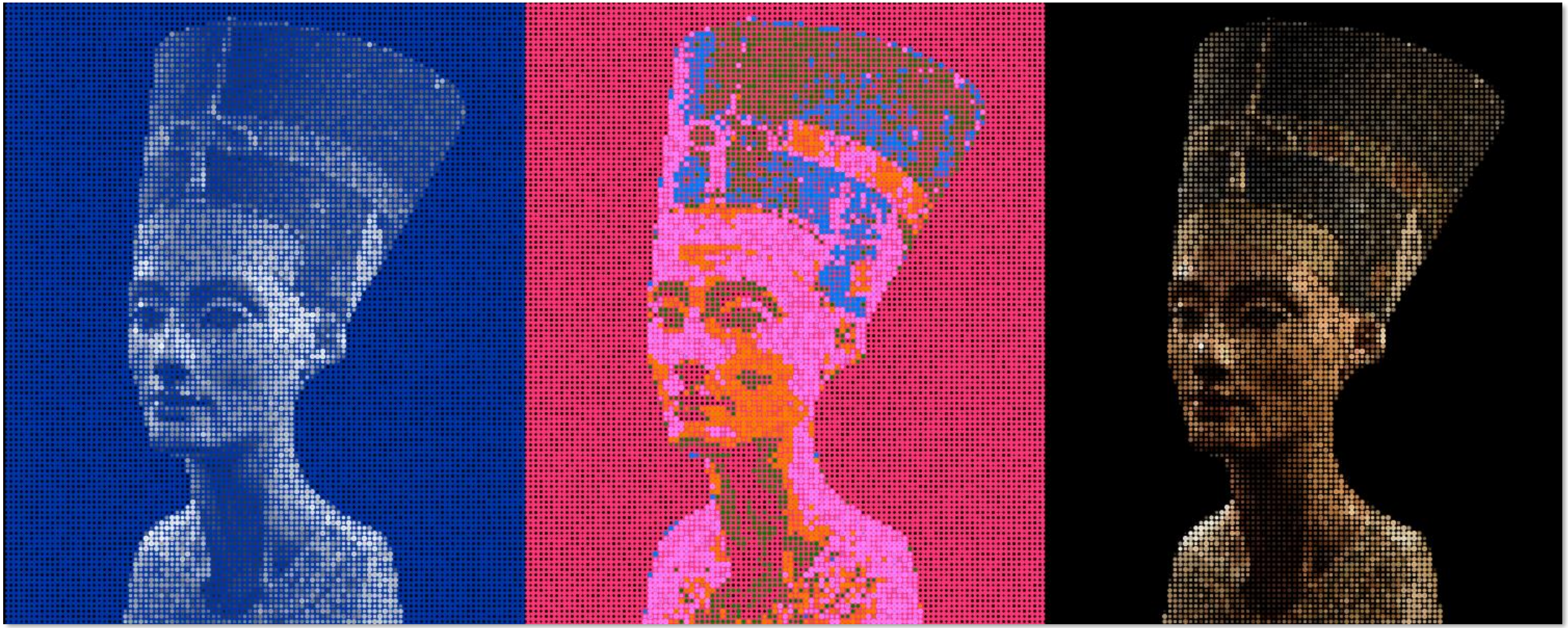


Световноизвестен поп арт художник, САЩ [2]

РЕАЛИЗАЦИЯ

ПРОЕКТ „НЕФЕРТИТИ В СЪВРЕМЕННИЯ СВЯТ“

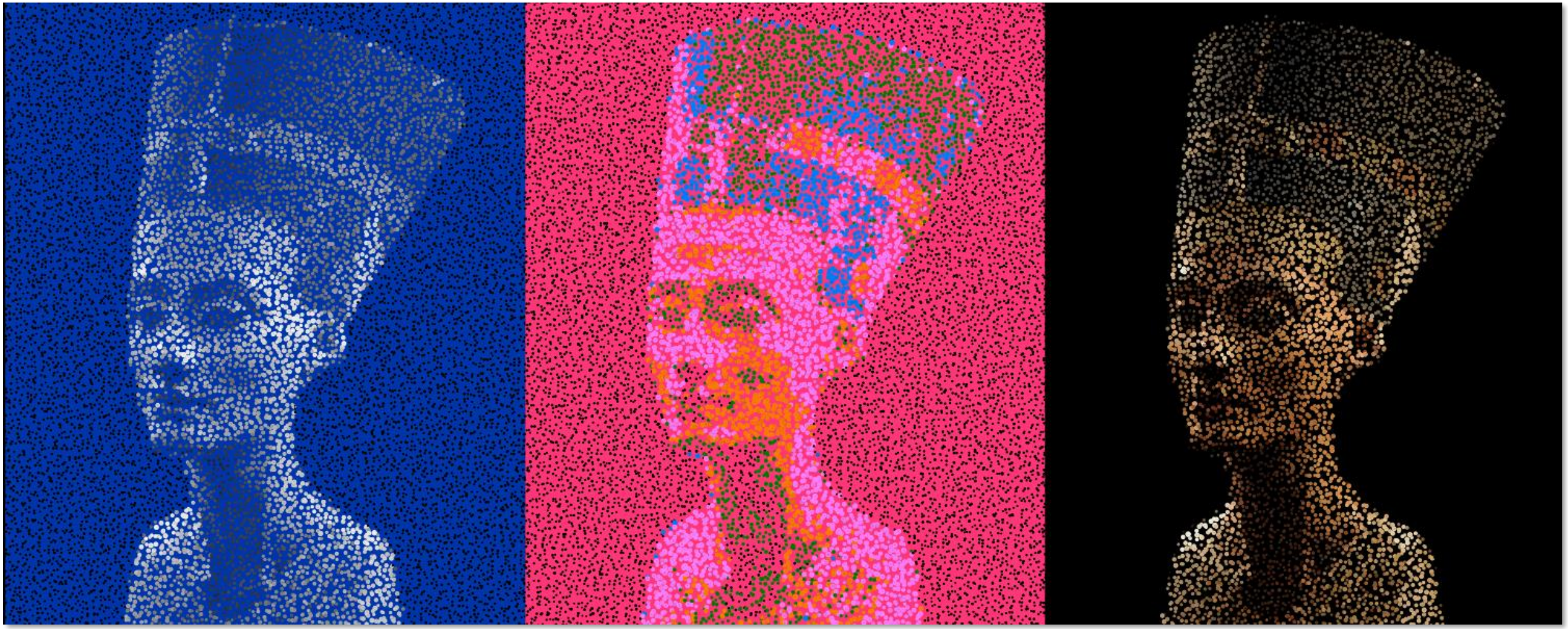
КОНСТРУКТИВНА ФАЗА



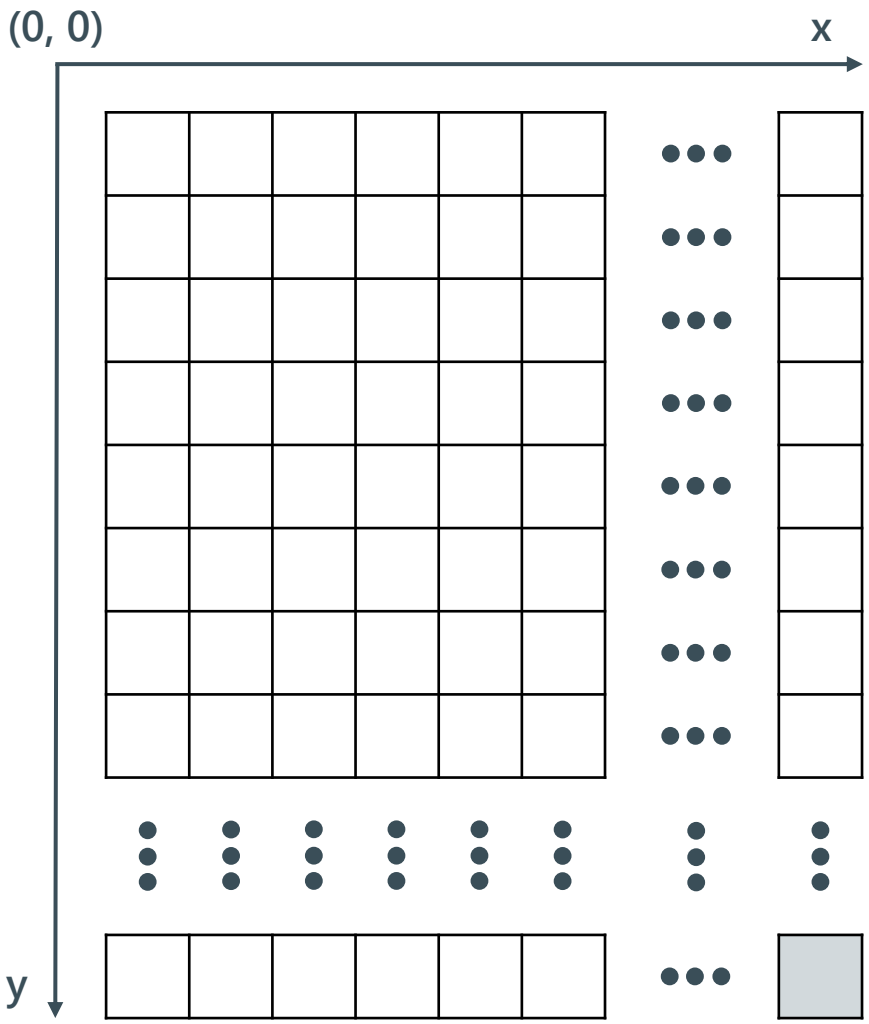
РЕАЛИЗАЦИЯ

ПРОЕКТ „НЕФЕРТИТИ В СЪВРЕМЕННИЯ СВЯТ“

ДЕСТРУКТИВНА ФАЗА



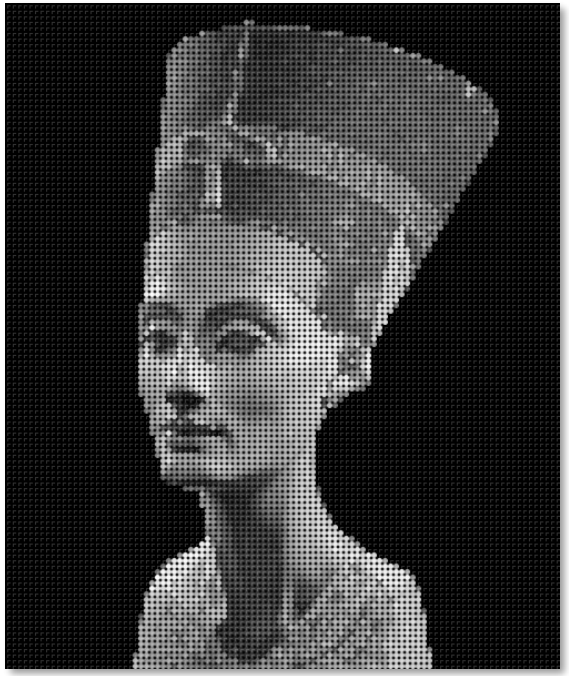
РЕАЛИЗАЦИЯ

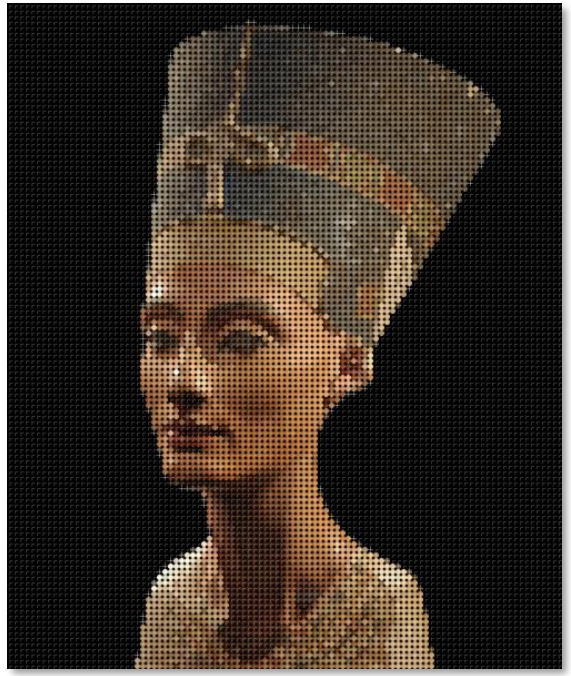


color c = img.get(imgX, imgY);

АЛГОРИТЪМ

ОСНОВА НА МОДЕЛА





// Black and white image.
fill(brightness(c));

// Full color image.
fill(c);

tiles = 300

[3]

РЕАЛИЗАЦИЯ

АЛГОРИТЪМ

СЛУЧАЙНОСТ

```
void drawTiles(PImage img, float offsetX, float offsetY, float tiles, float tileSize) {
  for (int x = 0; x < tiles; x++) {
    for (int y = 0; y < tiles; y++) {
      // Get pixel color from the current image.
      int imgX = (int)(x * tileSize);
      int imgY = (int)(y * tileSize);
      imgX = constrain(imgX, 0, img.width - 1);
      imgY = constrain(imgY, 0, img.height - 1);

      color c = img.get(imgX, imgY);
      float brightnessValue = brightness(c);

      // Refine scatter effect for clearer destruction phase.
      float scatterX = constructing ? 0 : random(-tileSize * 0.5, tileSize * 0.5);
      float scatterY = constructing ? 0 : random(-tileSize * 0.5, tileSize * 0.5);

      // Determine size based on construction or destruction phase.
      // Normalize time for smooth interpolation.
      float phase = map(time, 0, 120, 0, 1);
      phase = constrain(phase, 0, 1);

      float size = constructing
        ? lerp(0, tileSize, phase) // Gradually grow tiles during construction.
        : lerp(tileSize, 0, phase); // Gradually shrink tiles during destruction.

      // Scale size based on brightness for better detail.
      size *= map(brightnessValue, 0, 255, 0.5, 1.0);

      drawShadedEllipseBottomUp(x * tileSize + offsetX + scatterX, y * tileSize + offsetY + scatterY, size, c, 0); // Bottom-up recursion.
    }
  }
}
```

Във функцията *drawTiles()* по време на изпълнението на деструктивната фаза (construction == false) позицията (x, y) на всяка една точка е разпръсната случайно в произволна посока на базата на tileSize.

РЕАЛИЗАЦИЯ

АЛГОРИТЪМ

СЛУЧАЙНОСТ

```
void drawShadedEllipseBottomUp(float x, float y, float size, color c, int level) {
  // Base case to stop recursion once size is too small.
  if (size <= 1) return;

  // Randomly modify the size to add variety.
  float randomnessFactor = random(0.8, 1.2);
  float newSize = size * randomnessFactor;

  // Calculate shading based on new size.
  float gradientFactor = map(newSize, 0, size, 0, 1);

  // Draw the ellipse at the current level of recursion.
  fill(lerpColor(color(0), c, gradientFactor));
  ellipse(x, y, newSize, newSize);

  // Recursively call for the next level, reducing the size further.
  drawShadedEllipseBottomUp(x, y, newSize * 0.8, c, level + 1); // Recursive call to draw smaller ellipses.
}
```

Във функцията *drawShadedEllipseBottomUp()* независимо от вида на изпълняваната фаза размерът на всяка една елипса е случаен в рекурсивното извикване на метода.

map() – пренасочване на число от един диапазон към друг (в случая се използва за изчисление на градиентен фактор) [4]

$$mappedValue = min2 + \left(\frac{value - min1}{max1 - min1}\right) \times (max2 - min2)$$

lerpColor() – изчисляване на цвят между два цвята с определена стъпка на интерполация между двете стойности [5]

РЕАЛИЗАЦИЯ

АЛГОРИТЪМ

ИТЕРАЦИЯ

```
void drawTiles(PImage img, float offsetX, float offsetY, float tiles, float tileSize) {
  for (int x = 0; x < tiles; x++) {
    for (int y = 0; y < tiles; y++) {
      // Get pixel color from the current image.
      int imgX = (int)(x * tileSize);
      int imgY = (int)(y * tileSize);
      imgX = constrain(imgX, 0, img.width - 1);
      imgY = constrain(imgY, 0, img.height - 1);

      color c = img.get(imgX, imgY);
      float brightnessValue = brightness(c);

      // Refine scatter effect for clearer destruction phase.
      float scatterX = constructing ? 0 : random(-tileSize * 0.5, tileSize * 0.5);
      float scatterY = constructing ? 0 : random(-tileSize * 0.5, tileSize * 0.5);

      // Determine size based on construction or destruction phase.
      // Normalize time for smooth interpolation.
      float phase = map(time, 0, 120, 0, 1);
      phase = constrain(phase, 0, 1);

      float size = constructing
        ? lerp(0, tileSize, phase) // Gradually grow tiles during construction.
        : lerp(tileSize, 0, phase); // Gradually shrink tiles during destruction.

      // Scale size based on brightness for better detail.
      size *= map(brightnessValue, 0, 255, 0.5, 1.0);

      drawShadedEllipseBottomUp(x * tileSize + offsetX + scatterX, y * tileSize + offsetY + scatterY, size, c, 0); // Bottom-up recursion.
    }
  }
}
```

Във функцията *drawTiles()* два вложени for цикъла се използват за итериране на всички точки на мрежата върху основата на изображението, след което се взимат стойностите за локация и цвят.

constrain() – ограничаване на определена стойност в рамките на минимален и максимален лимит

lerp() – изчисляване на число между две числа с определена стъпка на интерполация между двете стойности (при определяне на размера на точките спрямо това дали фазата е конструктивна, или е деструктивна) [6, 7]

РЕАЛИЗАЦИЯ

АЛГОРИТЪМ

ИТЕРАЦИЯ

```
PImage convertToGrayscale(PImage img) {
  PImage grayImage = createImage(img.width, img.height, RGB);
  img.loadPixels();
  grayImage.loadPixels();

  // Convert each pixel to grayscale.
  for (int i = 0; i < img.pixels.length; i++) {
    color c = img.pixels[i];
    float gray = brightness(c);

    grayImage.pixels[i] = color(gray, gray, gray);
  }
  grayImage.updatePixels();
  return grayImage;
}
```

```
PImage applyPopArtColors(PImage img) {
  PImage popArtImage = createImage(img.width, img.height, RGB);
  img.loadPixels();
  popArtImage.loadPixels();

  // Apply pop art colors (strong contrasts).
  for (int i = 0; i < img.pixels.length; i++) {
    color c = img.pixels[i];

    // Apply basic thresholds to create bright contrasting colors.
    float r = red(c) > 128 ? 255 : 0;
    float g = green(c) > 32 ? 128 : 0;
    float b = blue(c) > 64 ? 255 : 0;

    popArtImage.pixels[i] = color(r, g, b);
  }
  popArtImage.updatePixels();
  return popArtImage;
}
```

Във функцията ***convertToGrayscale()*** с for цикъл се итерират всички пиксели на изображението, след което се конвертират в скалата на сивото в новото изображение. Аналогично при ***applyPopArtColors()***.

[3]

РЕАЛИЗАЦИЯ

АЛГОРИТЪМ

РЕКУРСИЯ

```
void drawShadedEllipseBottomUp(float x, float y, float size, color c, int level) {
  // Base case to stop recursion once size is too small.
  if (size <= 1) return;

  // Randomly modify the size to add variety.
  float randomnessFactor = random(0.8, 1.2);
  float newSize = size * randomnessFactor;

  // Calculate shading based on new size.
  float gradientFactor = map(newSize, 0, size, 0, 1);

  // Draw the ellipse at the current level of recursion.
  fill(lerpColor(color(0), c, gradientFactor));
  ellipse(x, y, newSize, newSize);

  // Recursively call for the next level, reducing the size further.
  drawShadedEllipseBottomUp(x, y, newSize * 0.8, c, level + 1); // Recursive call to draw smaller ellipses.
}
```

Във функцията *drawShadedEllipseBottomUp()* се използва рекурсия за прогресивно рисуване на елипси с по-малък размер (80% от текущия размер), докато не се достигне дъното на рекурсията. Ефект на засенчване с *lerpColor()*.

[5]

ЗАКЛЮЧЕНИЕ

Идея

ИСТОРИЯ НА ИЗКУСТВОТО

Симбиоза между културата на Древен Египет в лицето на Нефертити и поп арт стила от XX в.

Алгоритъм

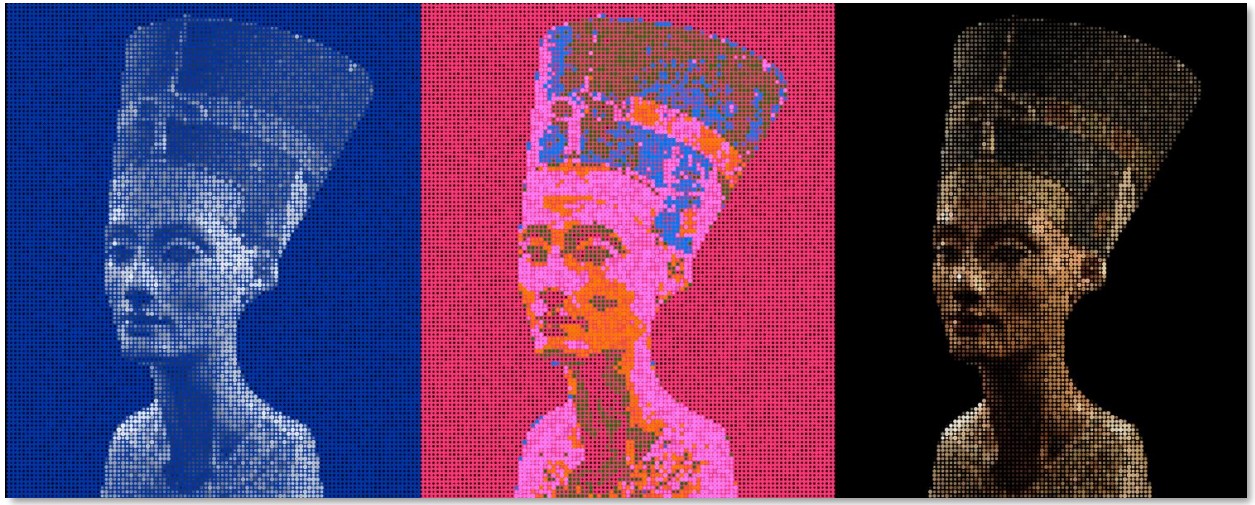
ОСНОВНИ КОМПОНЕНТИ

Имплементация на случайност, итерация и рекурсия в проекта „Нефертити в съвременния свят“

Резултат

ПРОЕКТ

Интерактивен проект в сферата на генеративното изкуство с преливане на фази на конструкция и деструкция



ЛИТЕРАТУРА

[1] Площад Славейков. (2015). *Нефертити – загадката на съвършената красота*. https://www.ploshtadslaveikov.com/nefertiti-zagadkata-na-savarshenata-krasota/, последно посетен на 12 януари, 2025 г.

[2] The Andy Warhol Museum. *Biography*. https://www.warhol.org/andy-warhols-life/, последно посетен на 13 януари, 2025 г.

[3] Kretzer, M. (2016). *Processing – Generative Design Tutorial: Image Mapping*. Digital Crafting. http://responsivedesign.de/wp-content/uploads/2016/05/tutorial-05_processing-imagemapping2.pdf, последно посетен на 11 януари, 2025 г.

[4] Map. *Processing*. https://processing.org/reference/map_.html, последно посетен на 14 януари, 2025 г.

[5] LerpColor. *Processing*. https://processing.org/reference/lerpColor_.html, последно посетен на 14 януари, 2025 г.

[6] Constrain. *Processing*. https://processing.org/reference/constrain_.html, последно посетен на 14 януари, 2025 г.

[7] Lerp. *Processing*. https://processing.org/reference/lerp_.html, последно посетен на 14 януари, 2025 г.

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!