**Generative Art Project**

This Python script uses the Pygame library to create generative art. The art consists of a cityscape-like scene with a sun, buildings, and grass-like elements. Each component is generated randomly, resulting in a unique and dynamic visual experience.

**Project Components**

**Display Parameters**

* **display\_size**: The size of the display window. By default, it's set to 100 pixels.
* **multiplyer**: A multiplier used to adjust the display size. In this script, it's set to 8.
* **hipo**: The length of the hypotenuse of a square with sides equal to **display\_size**. This value is used to normalize distances.
* **rect\_size**: The size of individual rectangles used for drawing elements in the scene.

**Sun Logic**

* **sun\_noise**: A list to store two random values used for sun logic.
* **sv1** and **sv2**: Randomly generated values to introduce noise in the sun's appearance.

The sun logic determines the color and position of the sun based on the distance from the center of the display. It creates a visual effect of a sun with varying colors.

**Building Logic**

* **building** and **building2**: Lists to store parameters of two buildings. Each building is represented by its starting and ending x-coordinates and its height.

The building logic is responsible for drawing buildings based on their positions. It applies noise to the building color and creates a layered effect for the buildings.

**Distance Functions**

* **distance(poit1x, poit1y, poit2x, poit2y)**: Calculates the Euclidean distance between two points.
* **line\_distance(poit1, point2)**: Calculates the distance between two points on a line segment.
* **average(x, y)**: Calculates the average of two values.

These functions are used to calculate distances between points, which are crucial for determining colors and positions in the scene.

**Pygame Initialization**

* **pygame.init()**: Initializes the Pygame library.
* **screen**: Creates the display window with the specified dimensions.

**Main Loop**

The script uses nested loops to iterate through each pixel in the display window. For each pixel, it calculates various parameters such as distance and applies logic for the sun, buildings, and noise.

**City Render Function**

* **city\_render()**: A function that contains the main logic for rendering the city scene. It is called within the main loop.

**Event Handling**

The script listens for a quit event (e.g., the user closing the window) and terminates the program accordingly.