Retro Snow

In this project, you will create a retro animation using terminal escape sequences, that mimics falling snow. This is, in fact, a trivial particle system. Sophisticated particle systems are a routine part of visual effects in television and film as well as crucial tools in scientific computing.

A Particle

A single particle looks like this:

```
struct Particle {
    int32_t line;
    int32_t column;
};
```

Thus, a particle is made up of two 32 bit integers side-by-side. The first provides the line of the particle, the second its column.

Screen Size

A default terminal window measures 80 characters across and 24 lines down. The origin is (1, 1) located at the top left corner.

Cursor Movement

Snow flakes (particles) will be positioned using an old school VT100 escape sequence. This sequence is:

```
ESC [ line ; column H
```

with no spaces.

Using `printf()`, a template string can be employed:

```
"\033[%d;%dH"
```

In C++, you'd do something like this:

```
cout << "\033[" << line << ";" << column << "H";
```

Note that there are no new lines being printed since this would in itself move the cursor to the left edge of the next line and could even cause scrolling - a disaster in this project.

Erasing the Screen

To erase the screen, use this escape sequence after first having moved to screen position (1, 1):

```
"\033[2J"
```

A C++ Object To Use As A Model

The following C++ can be used to start your thinking about how to go about writing this project.

```
struct Particle {
    int32_t line;
    int32_t column;
    void Step();
    void Render();
    void Reset();
};
```

`Step()`

All particles use the `Step()` function to move into the future. The `line` is increased by one and the `column` can either remain unchanged, move to the left or move to the right.

If the particle's `line` exceeds 24, the particle is `Reset()`.

`Render()`

To draw a particle, `Move()` to its location then print a "*" all without emitting a new line. Before printing the "*", check to ensure both the `line` and `column` are within the boundaries of the default terminal window (i.e. line between 1 and 24 and also column between 1 and 80). If the particle's position is outside this range, don't print anything.

`Reset()`

A particle's initial `column` can be anywhere from 1 to 80 inclusive. Special care is given to setting the particle's initial `line`. Specifically, always place the particle above the visible screen... chosen randomly from -1 to -48. This allows for sufficient spacing so that the snow does not appear to fall in clumps.

The Storm

A single snow flakes does not a blizzard make. Therefore, allocate a large number of them - perhaps 150 to 200 flakes. Since assembly language doesn't have actual C++ objects, each flake occupies only the 2 `int32_t` values. Suppose n is the number of flakes you choose to model. Then allocate $2 * 4 * n$ bytes using `malloc()`. Since you will then `Reset()` every one of them, initializing the newly allocated memory is not needed.

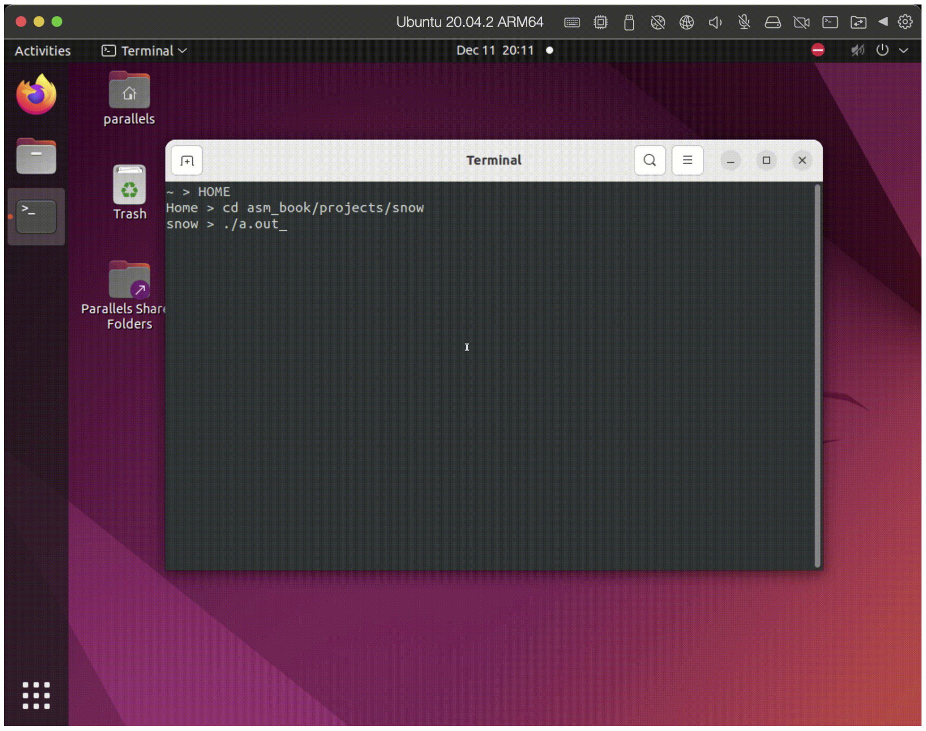
Then, you'll need a `StepAll()` and a `RenderAll()`.

The Main Loop

- Position cursor at (1, 1).
- Erase the screen.
- `StepAll()`
- `RenderAll()`
- Position cursor at (1, 1) again and print a new line.
- Delay a short time. In my implementation, I using `usleep()` to delay 217 microseconds.

Note only one new line is printed per frame - this is used to force all output to be flushed to the terminal.

Desired Outcome



```
~ > HOME
Home > cd asm_book/projects/snow
snow > ./a.out_
```

Figure 1: movie

Solution

The solution is here.