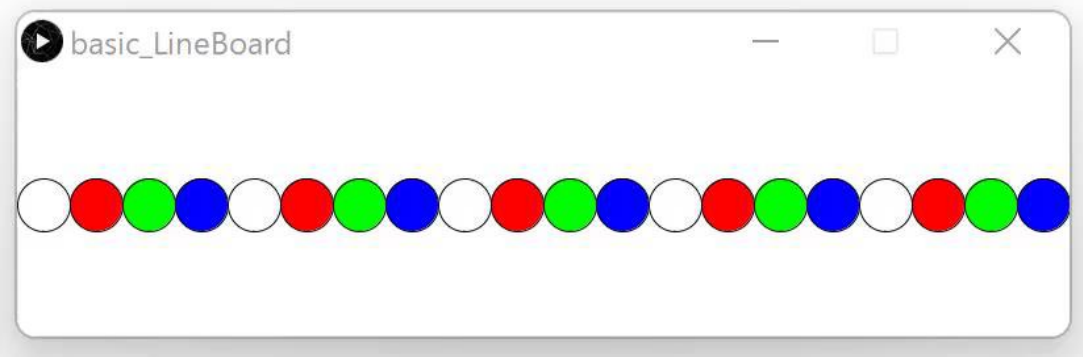
プログラミング演習I （第8回） 課題

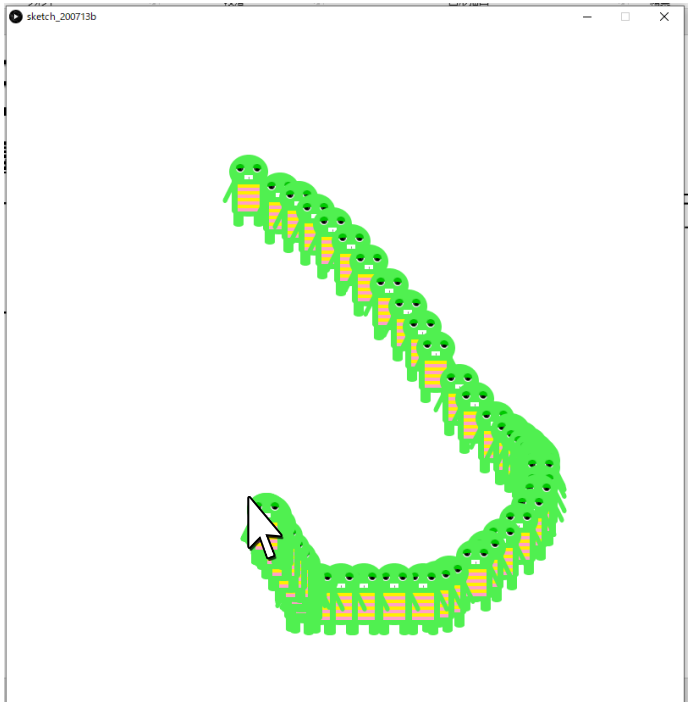
- **基本①　スケッチ名： basic_LineBoard**
  - 800x200のウインドウを作成し、直径40の円を横に等間隔に20個並べよ（円の中心のY座標は100とせよ）。また円の内部をクリックする度に、そのクリックされた円の色が【白→赤→緑→青→白（以後ループ）】と変化させるようにせよ。なお、起動したときの円の色は下記のようになるようにせよ。

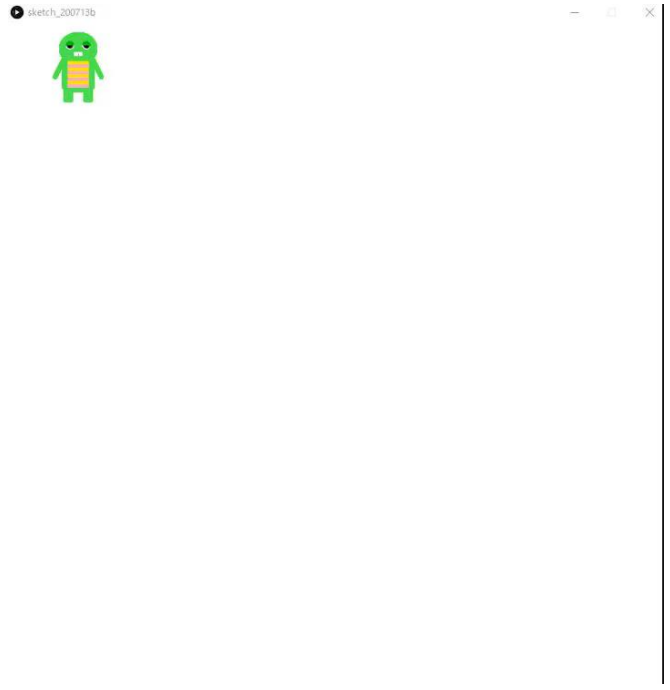


https://gyazo.com/438c9aefbcee009a7204cd2c299d9e6d

プログラミング演習I （第8回） 課題

- **基本②スケッチ名： basic_MouseTrace**
  - 800x800のウインドウを作成し，そのウインドウの中でマウスカーソルを動かすと，マウスカーソルを追尾する50個のキャラクタを描画せよ
  - ただし，マウスを追尾しているキャラクタのうち，マウスに近いもの（新しいもの）を，手前に表示するようにせよ
  - 最初に左上にキャラクタが集まっててもよい





キャラクタを小さくする

- drawCharacterを下記のように用意しよう

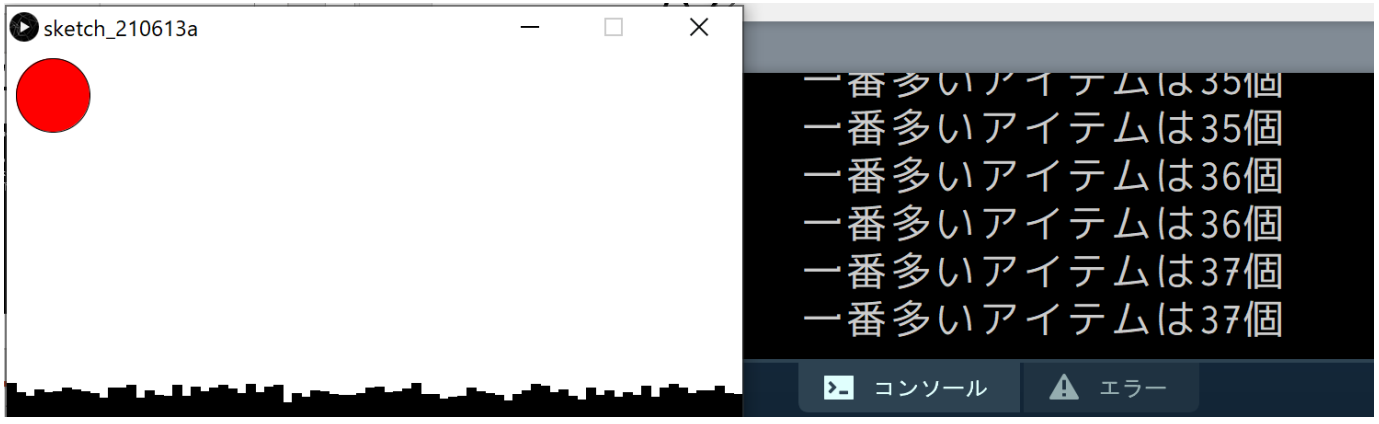
```
void drawCharacter(int cx, int cy)
{
  // scaleで使うための準備。以下の場合は0.2倍にする
  float fScale = 0.2;
  pushMatrix();
  translate(cx-250*fScale, cy-250*fScale);
  scale(fScale);
  // ここから

  ここにキャラクタのプログラムコピペ！


  // ここまで
  popMatrix();
}
```

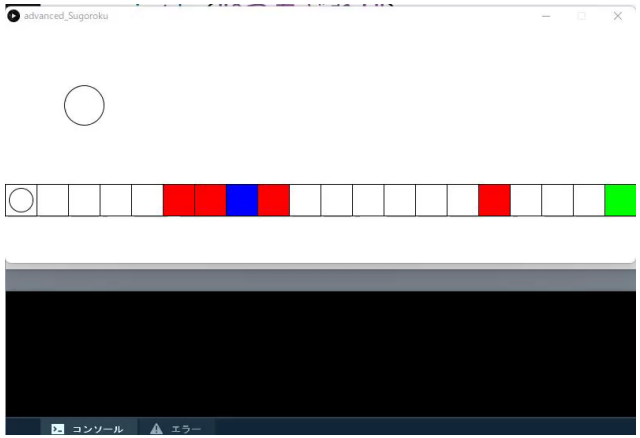
プログラミング演習I （第8回） 課題

- **基本③スケッチ名：basic_Gacha100**
  - 100個のアイテムがあり、画面左上に配置された直径80ピクセルの赤色の丸ボタンをクリックするたびに重複含め50個のアイテムをランダムに取得できる50連ガチャがある。ここで取得したアイテムの数をカウントし、その数を棒グラフとして表示するプログラムを作成せよ。ただし、配列を用いて実現せよ。
  - また、50連ガチャでアイテムを得た後に、各々のこれまで得たアイテム数を数え、最も多いものの数を標準出力せよ（累計で一番重複が多いアイテムの数を出力することになる）
  - なお、ウインドウの大きさは800x400とせよ。また丸ボタンの外をクリックした場合は、50連ガチャが発動しないようにせよ。



プログラミング演習I （第8回） 課題

- **発展① advanced_Sugoroku**
  - 0マス目にスタートがあり、19マス目にゴールがあるスゴロクを作成せよ。また、画面上に配置する何らかのボタンをクリックすることで1つのサイコロを振り、出た目だけ進むようにせよ（サイコロの出た目は標準出力するだけでも良いが画面に表示しても良い）。さらに、ゴールまでたどり着くと「Finish」と標準出力するか、画面上でお祝いするようにせよ。なお、ゴールのあとは行きすぎないようにして最後のコマにとまるようにせよ。
  - ただし、スゴロクにはそのマスに止まると戻るマス目（3つ戻る、スタートに戻るなど）と、何らかの進むマス目（3つ進む、5つ進むなど）を最低1つずつ配置するようにせよ。文字で示しても色で示しても良い。また配置数は好きにしてよし。



https://gyazo.com/d55603c5fc271297caca9d5d8cae50b5

プログラミング演習I （第8回） 課題

- **発展②　スケッチ名： advanced_Eratosthenes**
  - 次ページに示すエラトステネスの篩のアルゴリズムに従い，配列を利用して2～100万までの素数の個数を順次計算して求め，2～100万までの素数の個数を標準出力するとともに，計算を開始してから終了するまでの時間をミリ秒単位で計算し，出力せよ．この計算時間は100ミリ秒以内とすること．
    - こちらでもOK： https://ja.wikipedia.org/wiki/エラトステネスの篩
  - プログラムの出力例は下記のようにせよ

```
エラトステネスのふるいで組んだプログラム
2から100万までの素数は78498個
計算にかかった時間は13ミリ秒です
```

参考：エラトステネスの篩を配列で

1. 100万1個の要素からなるbooleanの配列を作る
   - 基本的な考え方は、前から順に数字を探索し、trueなら素数で、その倍数を全部素数じゃないと判定する。intでやってもいいよ！
2. 配列の全ての要素の値をtrueにする
   - ある数値に対応する値がtrueだったらその数は素数で、falseだったら素数ではないという考え方になる
3. numを2とする
4. 配列のnum番目の要素の値を確認する
   - trueだったら素数である
     - 素数の数をカウントアップ
     - numの倍数の配列の値（num*2やnum*3など）をすべてfalseにする（倍数は素数ではないため）。本当はnum*num以上の倍数のチェックで良いのだけど...
   - falseだったら素数ではないので何もしない
5. numの数を1増やし、numが100万になるまで4.に戻る
6. 素数の数を表示する

今日使うテクニック

millis()でミリ秒単位の経過時間を取得する

- アプリケーションが起動されてからの時間は millis() で取得することが可能なので、処理前と処理後の millis()の差分を求めることで、経過時間を取得することができる

```
int start = millis();

// なんか複雑な処理を色々する
// その処理が終わった

int end = millis();
println("経過時間は" + (end-start) + "ミリ秒");
```

今日使うテクニック

millis()でミリ秒単位の経過時間を取得する

- 色々と処理するときはこんな感じの書き方も

```
int iStartMillis;
boolean bFlagStart = false; // スタートしたかどうかのフラグ
void setup() {
  size(300, 150);
  fill( 0 ); // 文字色を黒色に設定
}
void draw() {
  background( 255 );
  if( bFlagStart ){ // スタートしていたら～
    text( millis()-iStartMillis, 20, 90 ); // 差分で経過時間を表示
  }
}
void mousePressed(){
  bFlagStart = true; // クリックされたらスタートフラグを立てる
  iStartMillis = millis(); // スタートの経過時間をセット
}
```