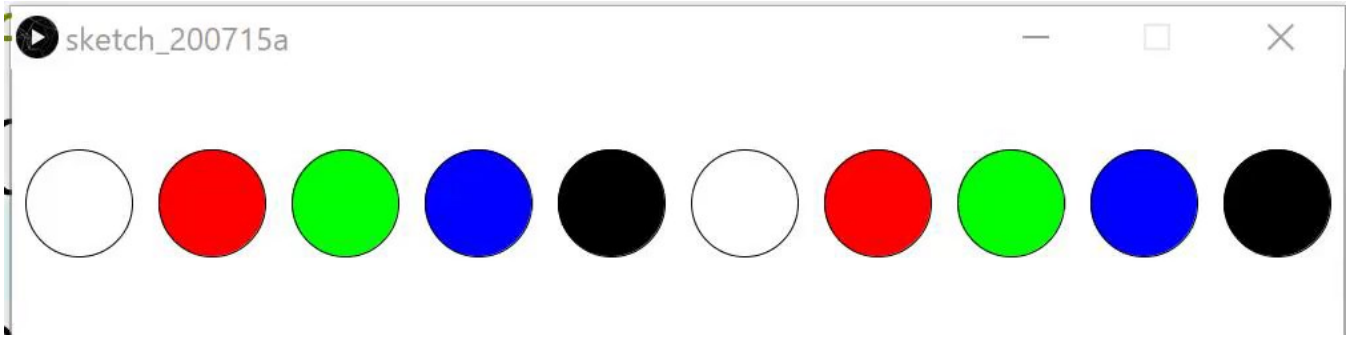
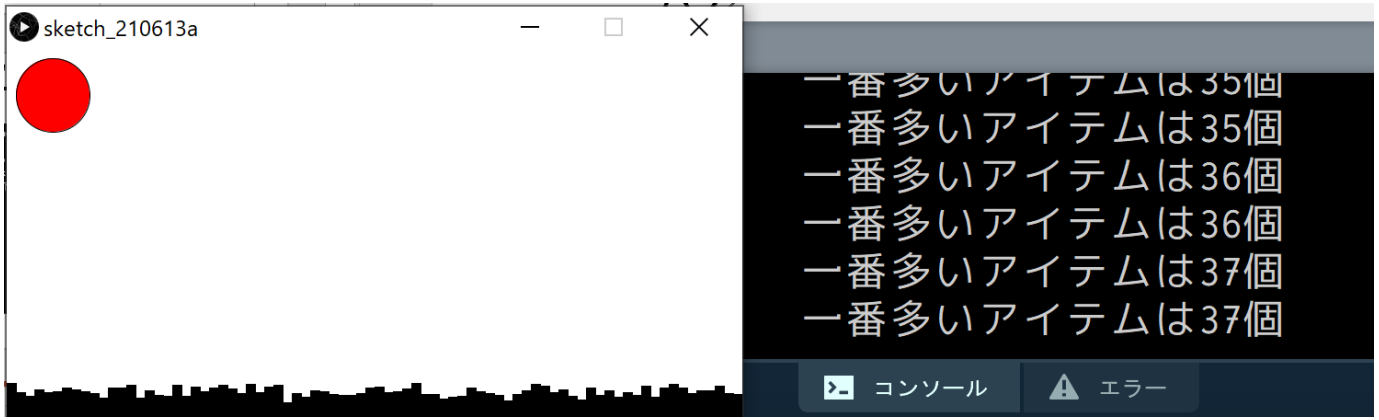
プログラミング演習I （第11回） 課題

- **基本① basic_LineBoard**
  - 500x100のウインドウを作成し、直径40の円を横に等間隔に10個並べよ（円の中心のY座標は50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→黒→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。



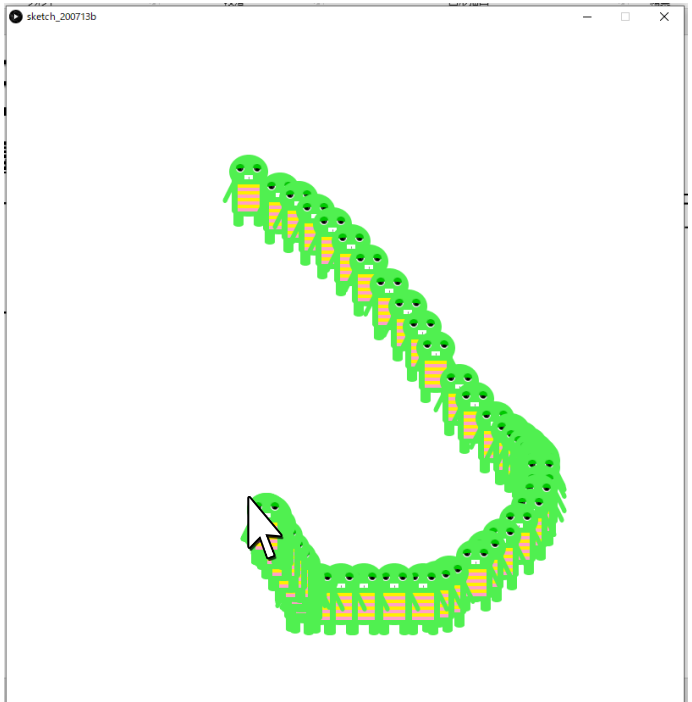
プログラミング演習I （第11回） 課題

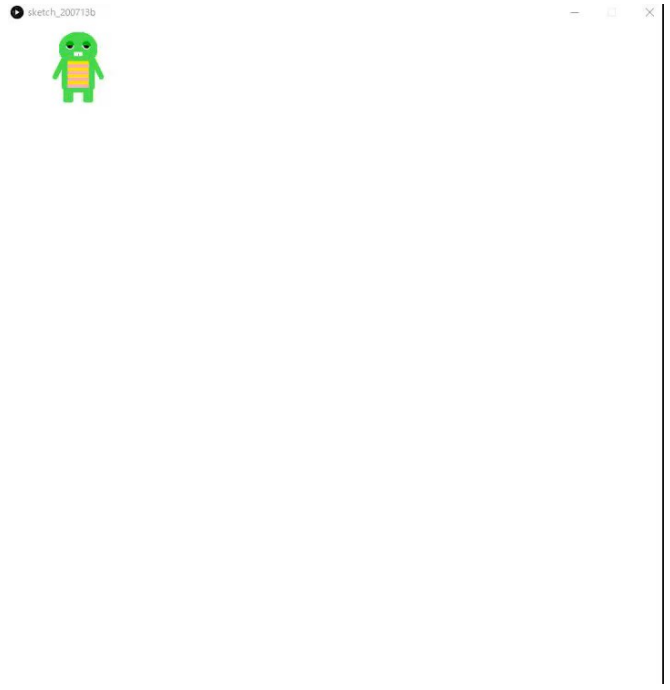
- **基本②スケッチ名：basic_Gacha100**
  - 100個のアイテムがあり、画面左上に配置された直径80ピクセルの赤色の丸ボタンをクリックするたびに重複含め50個のアイテムをランダムに取得できる50連ガチャがある。ここで取得したアイテムの数をカウントし、その数を棒グラフとして表示するプログラムを作成せよ。ただし、配列を用いて実現せよ。
  - また、50連ガチャでアイテムを得た後に、各々のこれまで得たアイテム数を数え、最も多いものの数を標準出力せよ（累計で一番重複が多いアイテムの数を出力することになる）
  - なお、ウインドウの大きさは800x400とせよ。また丸ボタンの外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11回） 課題

- **基本③ 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);
  pushMatrix();
  // 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();
  popMatrix();
}
```

プログラミング演習I （第11回） 課題

- **発展課題① advanced_Eratosthenes**
  - 単純な方法で2～100万までの素数の個数を求め，そ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計算時間をミリ秒単位で計算し，標準出力せよ（答えは78498個）
    - こちらは、かなり時間がかかっても良い
  - 次に，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，どちらがどちらなのかをわかりやすく表示するようにせよ

```
単純な方法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59511ミリ秒です

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
}
```

プログラミング演習I（第11回） 課題

- **発展②　スケッチ名：advanced_PlanesSeats**
  - ある飛行機には座席が264席あり、この飛行機に264人の乗客が乗る予定である。
  - ここで、1人目の客はチケットをなくして席番号がわからなくなってしまったため、適当な座席を選び勝手に座席に座った（ランダムに席を選定して座った）。
  - それ以降の客は、自身の席があいていればその席に座り、そうでなければ他の席を適当に選び勝手に座るものとする。
  - 最後の264人目が、自身の席に座れる確率は何%か。
  - 100,000回シミュレーションすることによりその確率を求めよ。
  - なお簡単のため、席番号は1番から264番まであり、1番目に乗り込んだ乗客の座席番号は1番、2番目に乗り込んだ乗客の座席番号は2番のように、乗り込んだ順に1～264番までの座席が割り当てられているものとして良い（もちろん完全なシミュレーションのため、座席もランダムに事前に割り当てても良い）。

出典:パズルの国のアリス 美しくも難解な数学パズル： 坂井 公, 斉藤 重之（日本経済新聞出版社）