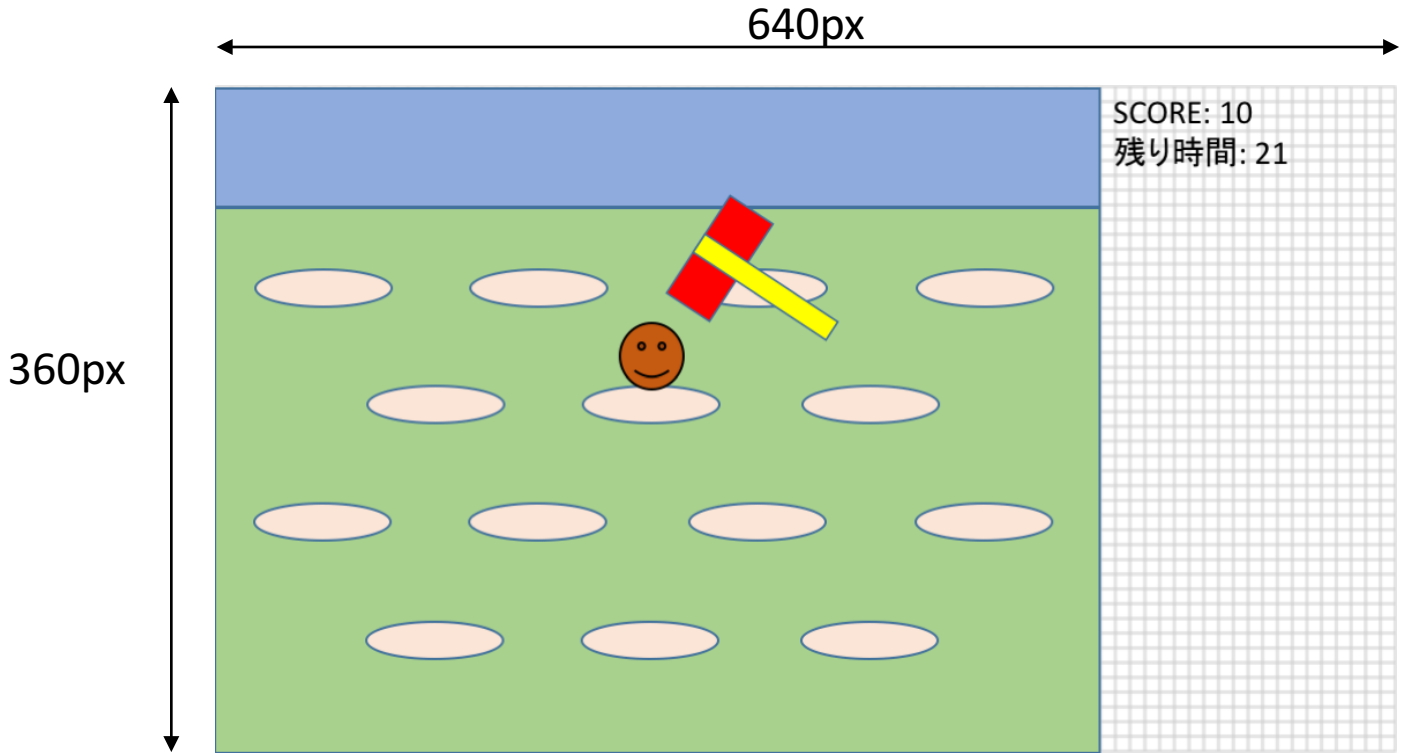
モグラたたき

企画:1

- 画面のラフ・スケッチ　※ラフ: rough だいたい



企画:2

- ゲームのメカニクス(機構)
  - 制限時間がある
  - モグラがランダムに出現する
  - ハンマーでモグラをたたくと
    - 得点が入る
    - モグラがランダムに移動する

企画:3

- 今回実装しないアイデア
  - 穴が固定された場所にあり、モグラは穴のところに出現する
  - たたいた効果を表示する
  - たたかないまま、ある時間経過すると、モグラが別な場所に移動する

プロトタイプ開発

※prototype 試作品

- いきなり本気で作らないで、試作してみる
- プロトタイプのメカニクス
  - 円がランダムな場所に出る
  - 円をクリックすると消えて、ランダムな場所に出る

ランダムな位置に出現

- 乱数random()を使用

接触判定

- 円形の当たり範囲
  - 簡単にするため
- 円の中心との距離で判定
  - 距離はdist()で求められる

プロダクト開発

※product 製品

- プロトタイプをベースに開発
- プロトタイプに追加するもの
  - 画像
  - 得点
  - 制限時間

画像の用意

- ステージ、モグラ、ハンマーを用意

得点

- 得点を表す変数を用意
- 得点を表示
- たたいたら、値を増やす

制限時間

- 現在の時間と制限時間を表す変数を用意
  - 残り時間 = 制限時間 − 現在の時間
- 残り時間を表示
- 制限時間を越えたら、ゲームを止める
  - 「GAME OVER」と表示してもいいかも

わかりにくいプログラムとは

- 「木を見て、森を見ない」
  - プログラムを素直に書くと、たくさんの木が並んでいるようなものになる
  - 一本一本の木を見るように、最初から最後まで読まないと、何をしているのかわからない
  - そして、人間は、一度にたくさん頭に入らないので、迷う

プログラムをわかりやすくするには

- 一部分を読んでも、わかるようにする
  - 一度にたくさん頭に入らない「人間にやさしい」
- 具体的には、「森」と「木」に分ける
  - 「森」には、大雑把な流れが書いてある
  - 「木」には、それぞれの木のことが書いてある

プログラムを部品にするオブジェクト指向

- オブジェクト指向
  - プログラムを部品として使う
- 今回、具体的には
  - 「ハンマー」「モグラ」などのオブジェクトを作る
  - メイン・プログラムには、それを使ったシナリオを書く

Javaのオブジェクト指向

- クラス(設計図)を宣言
  - フィールド(データ)とメソッド(操作)を用意
- インスタンス(実体)を、クラスから作って使う
  - インスタンスは、オブジェクト

ハンマー・クラスの作成

- フィールド: データ
  - 画像
  - x、y座標
- メソッド: 操作
  - 移動
  - 表示

ハンマー・クラスのコンストラクタ

- 初期化するときに行いたいことをコンストラクタに書く
- 具体的には
  - 画像の設定

カプセル化

- カプセル化とは
  - 本当は、外部から行っていい操作だけを見せて、そうでないものは隠すことが望ましい
  - private、protected、publicなどの修飾子の活用
  - 一方で、プログラムが長くなる
- 今回は、カプセル化は行わない
  - 開発時間と安全性のトレードオフ

※tradeoff: 代償

DRY: Don't Repeat Yourself
同じことを何度も書かかない

- ハンマーとモグラのプログラムは、似ている
- 親クラスを作って、共通の部分を吸収する

スプライト・クラス

※Sprite: 妖精

- ゲーム開発では、キャラクターなどの画像オブジェクトはスプライトと呼ばれている
- クラスの設計
  - フィールド
    - 画像、x、y座標
  - メソッド
    - 移動、表示、接触判定

ステージ・クラスの作成

- メイン・プログラムの整理
  - メイン・プログラムがシナリオだけに見えるように、ステージ・クラスを作って吸収
- 継承の限界
  - ステージ・クラスもスプライト・クラスを継承させる
  - あんまり共通なところがないので、継承しなくてもいいかも