最終レポート

＜メンバー＞

・5412086 湯浅日夏子
・5412070 鈴木智紗子
・5412035 能登瞳

＜企画コンセプト＞

買い物シュミレーションゲーム

＜作品の目的＞

小学生が頭を使いながら楽しく計算できるゲーム
消費税が 8%になったので、8%の面倒な消費税計算も身につく！

＜作品構想＞

・目的は消費税計算だが、それまでの買い物時の計算や予算を決定することで、より計算する場面を増やす
・飽きが来ないように、難易度の違うゲームを用意する

＜作品の独創的な点＞

・消費税が 5%から 8%に増税した今、最新の消費税計算
・指定された商品を買い物かごに入れていく既存のゲームに計算することを加えた。商品の指定だけでなく予算を設け、商品の合計金額、その金額に対する消費税を計算し、予算を上回らないよう、また予算をぎりぎりまで使うような商品選びをすることで計算力を高める

＜プログラムの仕様＞

クラス…Main クラス、Cost クラス、Decide クラス、Item クラス、Kago クラス、
　　Level クラス、Time クラス

「Cost クラス」

・decideCost
　受け取った番号に対応する商品の値段を返す

「Decide クラス」

・First

すでに決まっている、並べる商品番号の中に
引数として受け取った商品番号が被っているか否かを判定する。
引数として受け取った商品番号 a と、すでに決まっている並べる商品番号を
順番に比べていき、被る数字があればその時点で for 文を抜ける。
for 文を抜けた際に、繰り返しの回数をカウントする変数 k が、
引数 n と等しいとき、それは for 文を break せずに終えたということであり、
a は、すでに決まっている並べる商品番号の中には存在しないということを表す。

・decideRandom

引数 start 以上、end 未満の自然数をランダムで生成する。
しかしこの生成した値が、すでに決まっている並べる商品番号と被ってしまっては
いけないので、First メソッドを使用して判定しながら最終的な値を決定する。
ランダムで生成した値 a が、すでに決まっている並べる商品番号の中に存在したら
違う値を生成しなければならない。新しい値を生成する度に First メソッドを使って
すでに決まっている並べる商品番号と被らないかを判断し、被る番号がある限り
何度でも新しい値を生成し続ける。そうして最終的に決定した a の値を返す。

「Item クラス」

メンバー変数 is0, n0, cost0 をもつ。
このクラスのインスタンスを各商品につき 1 つずつ、計 9 つ作ることによって、
それぞれのインスタンスで
・その商品が表示されているか
・その商品の商品番号
・その商品の値段
を管理することができる。

・displayItem

受け取った野菜番号の野菜の画像を、受け取った引数 x と y を座標として表示する。

「Kago クラス」

クリックされた商品の商品番号を順に配列に格納していくクラス。
まずコンストラクタで配列の中身を全て 0 で初期化する。

・intoKago

配列の i 番目に受け取った商品番号 n を格納する

・drawItem

受け取った野菜番号の野菜の画像を、受け取った引数 x と y を座標として表示する。

「List クラス」

・decideList

受け取った引数 start 以上、end 未満の自然数をランダムで生成し買い物リストの１つ目とする。

２つ目の買い物リストも同じようにランダム(奇数のみ)で生成するが、買い物リストの１つ目と被ってはいけないので

while 文を使って、１つ目と被っている限り新しい値を生成し続ける。

買い物リストの３つ目も同様にして決める。

買い物リストに入っている商品をクリックすると買い物リストのその商品の欄が CLEAR になるが

全部クリアボタンを押すと CLAER と書いてあるところも全部元通りにしたい。

そのために、買い物リストに載る商品番号を決定した時点で２つの変数に格納しておく。

・displayList

決定した買い物リストの商品番号と受け取った値 x と y を引数として whatVeg を呼び出す。

・whatVeg

受け取った商品番号に対応する商品の名前を、受け取った引数 x と y を座標として表示する。

「Time クラス」

・decideTime

受け取った引数 time をメンバー変数 rest に格納し、このゲームの制限時間とする。

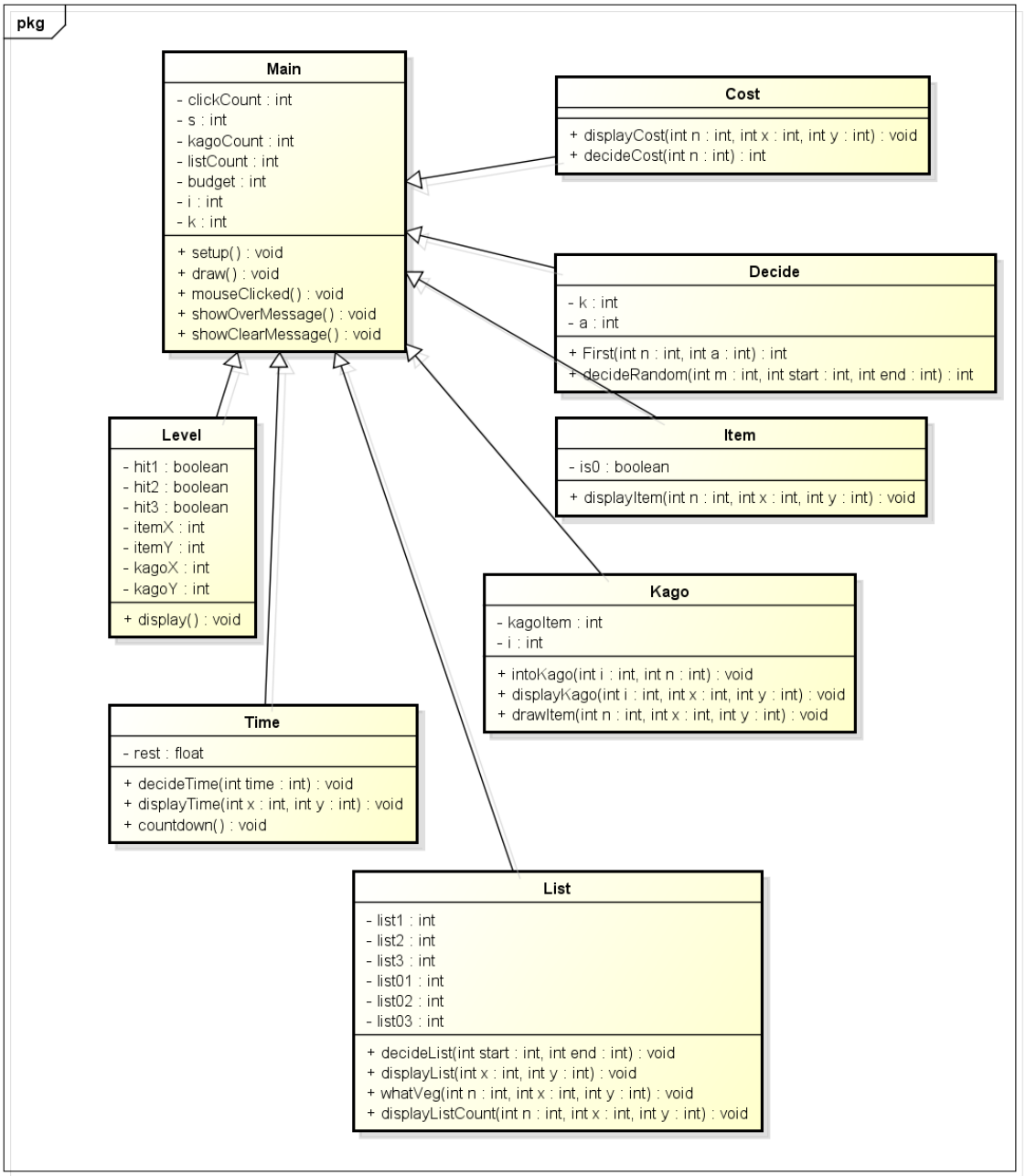
・displayTime

残り時間を表す rest を、受け取った引数 x と y を座標として表示する。

・countdown

呼び出される度に rest の値を 0.1 減らす。今回は frameRate(10) と指定してあるので

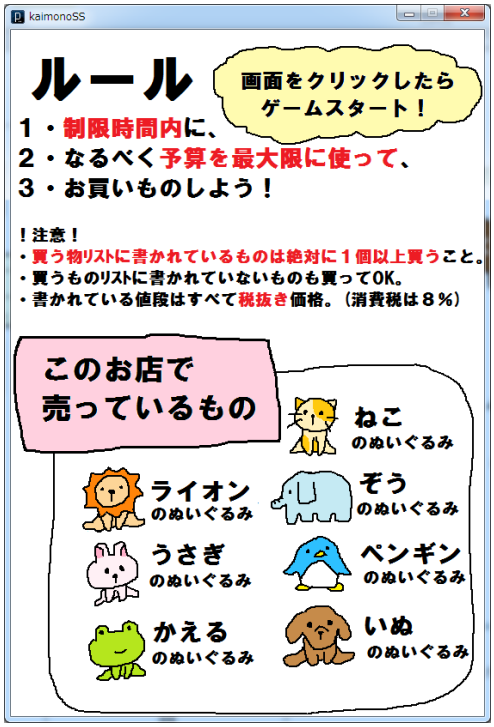
１秒につき 10 回このメソッドが呼び出され、ちょうど１秒につき rest の値が１小さくなる。


pkg
Main
- clickCount : int
- s : int
- kagoCount : int
- listCount : int
- budget : int
- i : int
- k : int
+ setup( ) : void
+ draw( ) : void
+ mouseClicked( ) : void
+ showOverMessage( ) : void
+ showClearMessage( ) : void
Cost
+ displayCost(int n : int, int x : int, int y : int) : void
+ decideCost(int n : int) : int
Decide
- k : int
- a : int
+ First(int n : int, int a : int) : int
+ decideRandom(int m : int, int start : int, int end : int) : int
Item
- is0 : boolean
+ displayItem(int n : int, int x : int, int y : int) : void
Level
- hit1 : boolean
- hit2 : boolean
- hit3 : boolean
- itemX : int
- itemY : int
- kagoX : int
- kagoY : int
+ display( ) : void
Kago
- kagoItem : int
- i : int
+ intoKago(int i : int, int n : int) : void
+ displayKago(int i : int, int x : int, int y : int) : void
+ drawItem(int n : int, int x : int, int y : int) : void
Time
- rest : float
+ decideTime(int time : int) : void
+ displayTime(int x : int, int y : int) : void
+ countdown( ) : void
List
- list1 : int
- list2 : int
- list3 : int
- list01 : int
- list02 : int
- list03 : int
+ decideList(int start : int, int end : int) : void
+ displayList(int x : int, int y : int) : void
+ whatVeg(int n : int, int x : int, int y : int) : void
+ displayListCount(int n : int, int x : int, int y : int) : void
powered by Astah

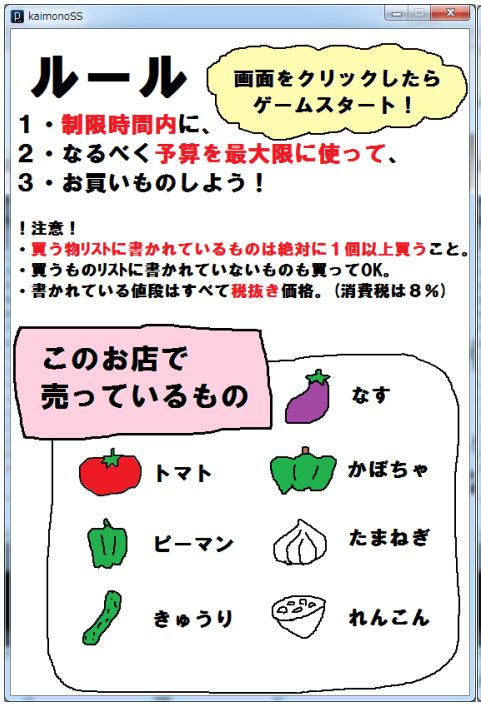
＜プログラムの実行の様子＞

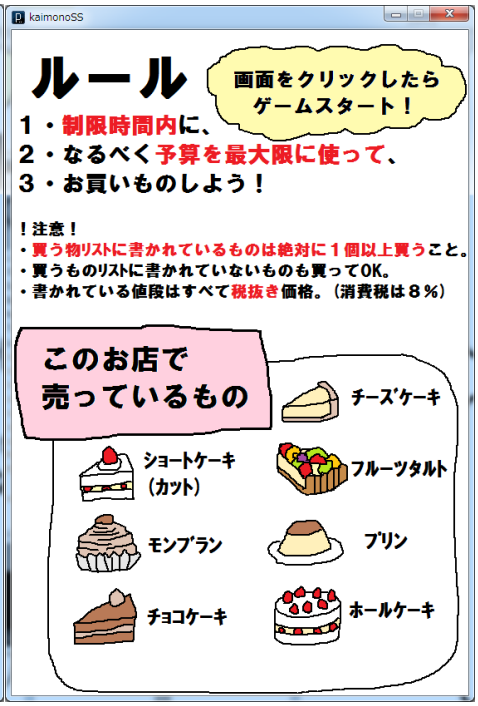
・オープニング画面

・ぬいぐるみ屋さんをクリックしたとすると、ルール説明画面になる。画面をクリックしてゲームスタート！



・八百屋さんとお菓子屋さんのルール説明画面





・ゲームが始まると、この画面に移る。

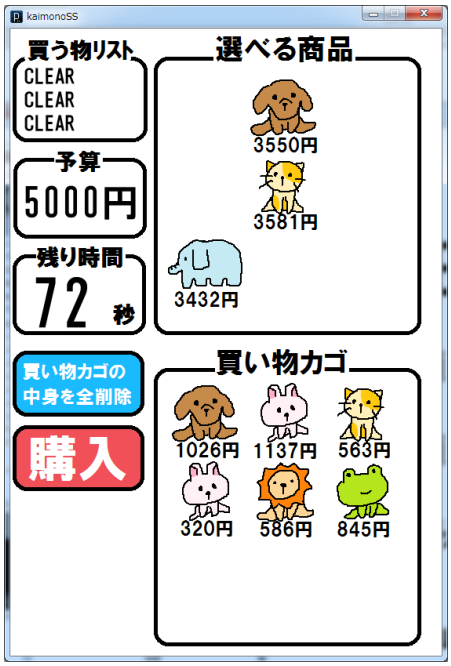
・買いたい商品をクリックするとその商品が買い物カゴに移動する。
　買う物リストの商品を購入すると、買う物リストのその商品の文字が[CREAR]に変わる。

・もし購入に失敗したと思ったら、青色の「買い物カゴの中身を全削除」ボタンをおすと、商品がリセットされる。

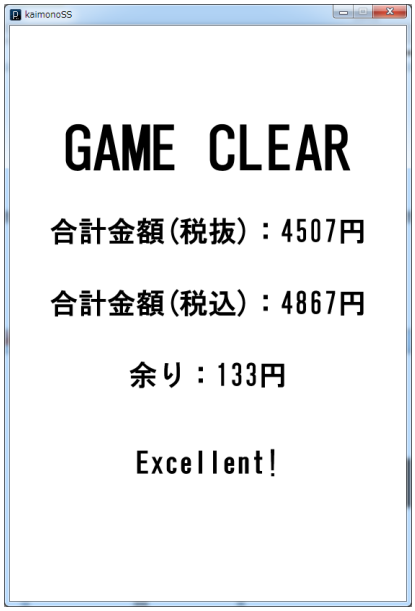
・買う物リストの商品を全部購入したので、赤色の「購入」ボタンをクリック。

・しかし、予算オーバーしていたのでゲームオーバー。”予算オーバーです”と表示される。

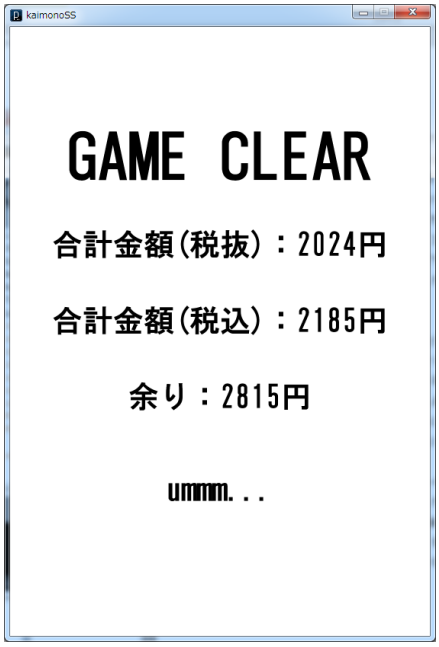
・買う物リストをすべて購入し、予算を超えないように、そして制限時間内に購入ボタンをクリック。



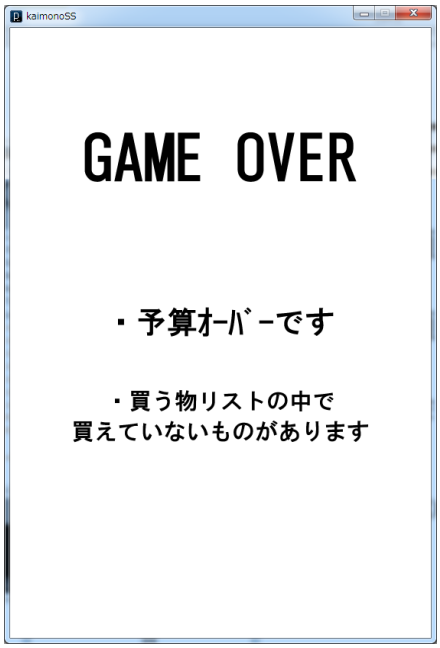
・買い物リストがそろえられたとき、制限時間内、予算内に購入できたときの 3 つの条件をすべてクリアすればゲームクリア。合計金額()、合計金額()、余りが表示され、今回は予算の 9 割を超えて購入できたので「Excellent!」と表示される。



・ゲームクリアしても、予算と合計金額の誤差によって表示される文字が異なる。今回は、もうちょっと頑張ろうという気持ちを込めて「ummm...」

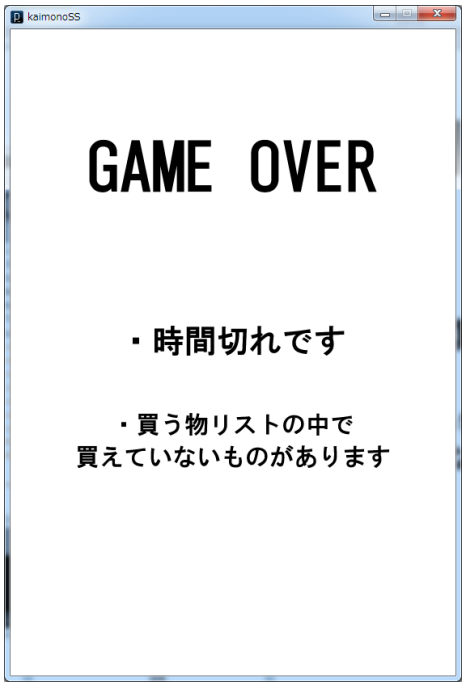


・クリア条件 3 つのうちひとつでもクリアできていなかったらゲームオーバー。今回は、制限時間内に購入ボタンをクリックしたものの、予算をオーバーしていて、買う物リストのものをコンプリートできていなかったときの表示画面。



・制限時間内に購入ボタンをクリックし、予算内に合計金額をおさめたものの、買う物リストがコンプリートできていなかかったときの表示画面

・予算内に購入していたものの、制限時間内に購入ボタンをクリックできず、買う物リストもコンプリートできていなかった時の表示画面



＜試用結果＞

一番苦労した点は、「買う物リスト」と「選べる商品」をランダムに表示すること。「買う物リスト」に表示されたものが必ず「選べる商品」にも表示されるようにすることだ。まず私たちがつまずいたのは、「買う物リスト」ありきで「選べる商品」を表示させるのか、「選べる商品」ありきで「買う物リスト」を表示させるか…。私たちは、先に買う物リストをランダムに 3 つ選ぶようにした。その際、その 3 つがすべて異なるような仕様にした。そして、その選ばれた 3 つが必ず 2 つずつ「選べる商品」に表示させることで、2 つの値段の異なる同じ商品のどちらを選択すれば、うまく買い物できるかユーザーに考えさせ、より頭を使うゲームになるようにした。

＜今後の課題＞

今回の最終的な画面には、予算と購入金額の誤差の大小によって「Excellent!」や「Great!」、「Good!」「ummm...」と表示させるようにした。だが、実際にはいろいろな案があった。

・プレイヤーは最初から持ち点があり、誤差を持ち点から引く。持ち点がある間はゲームを何度も繰り返しできるが、持ち点がなくなったらゲームオーバーでゲーム終了。

・プレーヤーには買い物するために必要な商品券が 5 枚あたえられ、ゲームできる回数は 5 回と決まっている。15 分経つと 1 枚ずつ商品券が支給されゲームできるようになる。

など…。

今回は時間とプログラムの問題から異なる文字を表示させるという仕様にしたが、もっと凝ろうと思えばいくらでも発展したゲームが作れると思った。

また、今は商品 1 つ 1 つの値段は絵の画像に書き込んでいるものなのでそれぞれの商品に 2 パターンずつしかない。だから、値段もすべてランダムに表示させるようにすれば値段のパターンも増えて、飽きることがなくなると思う。