デジタルコンテンツ 1
最終レポート

チーム名：10 トリス

作品名：10TRIS

メンバー

5412003 飯田佳徳
5412025 下田圭介
5412039 丸山隆太
5412079 疋田暁大

- 作品の狙い・アイディア

　我々のチームは本作品を制作するにあたり,子どもからお年寄りまで幅広い年齢層をユーザーの対象とすること,子どもには正確かつ素早い計算能力の向上と長い間計算を続けるための集中力,お年寄りには指先を使うことと単純な計算をすることによる脳の活性化を目的とした.そのために考えだされたものがテトリスのような要素を持つゲームである.テトリスのゲーム要素を取り入れた理由は,テトリスは多くの人がやったことのあるゲームでもあり,また操作が単純であるからやり続けられるという要素もあると思えたためである.

　そして,数字が割当てられたブロックがいくつか繋がって,さまざまな形で落下してくるものを移動させ,縦もしくは横で5個並べた時点でそれらの数字の和が10となるように計算を行わせるゲームを制作することとした.このような仕様とすることにより,移動する数字を即座に計算してある決められた値とするための計算能力、難易度の上昇により長い時間のあいだ計算をするための集中力、上下もしくは左右の計算を行うための素早い頭の回転を必要とするなどの目的を達成できると考えたためである.

- プログラムの仕様

1. number_tetris (メイン)
   keyPressed 関数
   – ゲームで利用されるキーが押されたときに各動作を司っているクラスのメソッドを呼び出す.
   display_gamestart 関数
   – ゲームのスタート画面の表示.
   display_gameover 関数
   – ゲームオーバーとなったときの画面の表示.

2. Field class
   draw_field メソッド
   – ブロックを描く
   clear_lines メソッド
   – 一行の総合値が 10 のときその一行を消す

3. Block class
   move_block メソッド
   – ブロックを一定の早さで下へ動かす
   move_block_down メソッド
   – ブロックを関数が呼び出された瞬間下へ動かす
   move_block_right メソッド
   – ブロックを今いる位置から右へ移動
   move_block_left メソッド
   – ブロックを今いる位置から左へ移動
   lock_block メソッド
   – 壁か、ブロックに到達したら現在動いてるブロックを固定する
   turn_block メソッド
   – 反時計回りにブロックを回転させる。

4. Block_check class
   create_block メソッド
   – ブロックを作成する
   check_block_down メソッド
   – 下へ壁か、ブロックがあるかチェックする
   check_block_right メソッド

– 右に壁か、ブロックがあるかチェックする
check_block_left メソッド
– 左に壁か、ブロックがあるかチェックする
check_block_turn メソッド
– 回転させようとするブロックの場所に壁か、ブロックがあるかチェックする

5. Status class
   express メソッド
   – スコア、ボーナス、レベル右画面に表示させる。
   score_cast メソッド
   – スコアに応じたレベルの表示、スコアに応じてボーナスを加える。

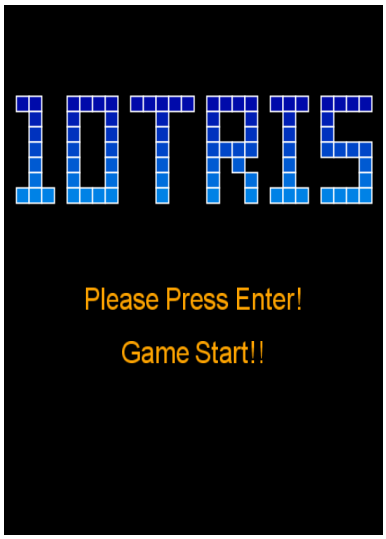
6. Level class
   level_decition メソッド
   – スコアによってレベルを変えてブロックの落下スピードを上げる。

7. Bonus class
   bonus_clear メソッド
   – スペースキーを押すと一番下の行のブロックを消去。
   bonus_clear_number1 メソッド
   – 1,2,3 のキーを押すとその値が描かれているブロックを消去。

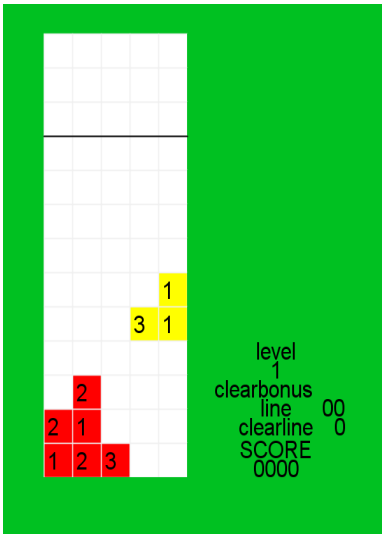
8. Back class
   start メソッド
   – スタート画面の描画.

9. Restart class
   reset メソッド
   – リスタート時にそれまでのステータスをすべて初期化する.

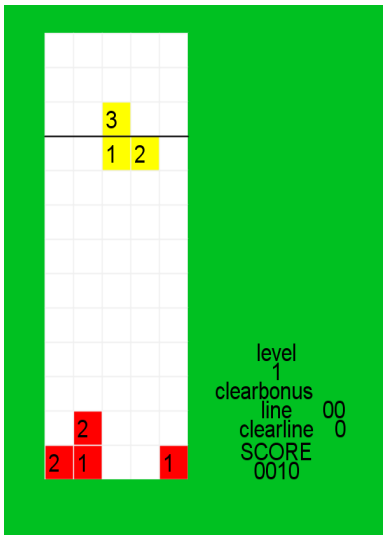
- プログラム実行の様子



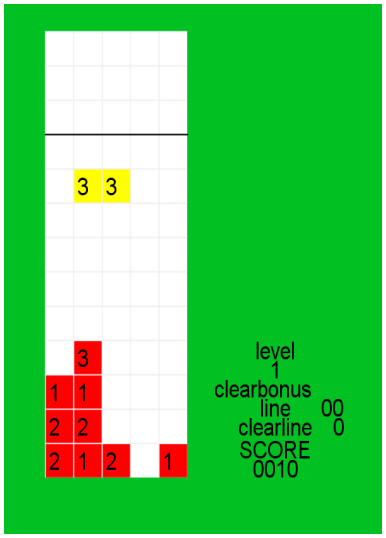
1. スタート画面
   Enter キーを押すことにより,ゲームスタート



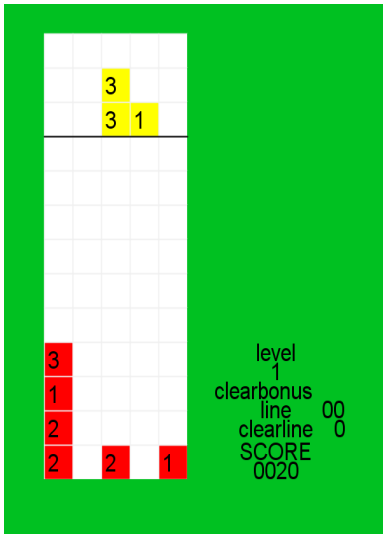
2. プレイ画面 (横でブロックを消す)
上から数字の割り当てられたブロックが落下してくる.
落下してくるブロックは黄色であり,操作できるようになっている.
赤いブロックは現在落下してきているブロックよりも前に落下してきたブロックであり,固定されている.
操作できるブロックを動かすことにより,縦もしくは横で 5 個のブロックを並べる.和が 10 となると,その部分の 5 個のブロックが消えるようになっている.
画像の最下段に注目すると,3 つのブロックが固定されており,その和は 6 となっている.そして,落下してきているブロックに着目すると,3 と 1 のブロックが繋がって落下してきているので,最下段の空いている部分に 3 と 1 のブロックを置くと和が 10 となり消える.



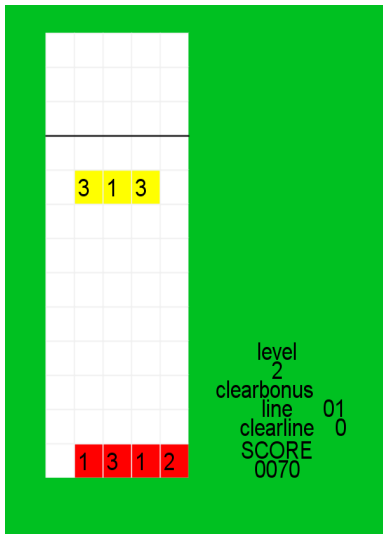
3. プレイ画面 (ブロックが消えた後)
ブロックが消えると,消えた段よりも上にあったブロックは消えた段数分(複数段同時に消すことも可能)だけ下へ移動する.
そして,一度消すとスコアが 10pt 加算される.
ただし,複数段同時に消してもスコアに加算されるポイントは 10pt のみである.



4. プレイ画面 (縦でブロックを消す)
画像で左から 2 列目に着目すると,縦に4つのブロックが積み重なっており,それらの和は7となっている.
そして,落下してきているブロックは,3 と 3 が繋がったブロックであるため,着目した列の上に 3 のブロックを置くと,和が 10 となって,その列のブロックは消える.

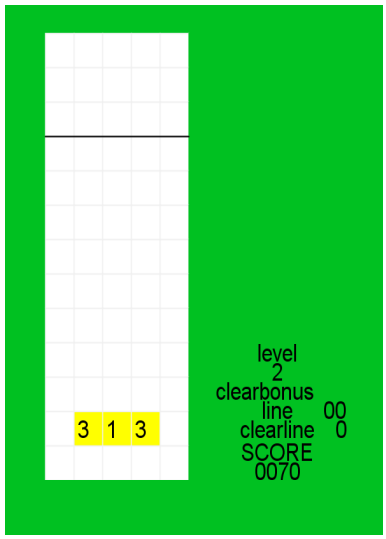


5. プレイ画面 (縦でブロックを消した後)
今回の場合はちょうど縦で5個しか並べなかったので,5個すべてのブロックが消え,左から2列目の場所には,何のブロックもない.しかし,消した列の上にブロックがある場合には,横でブロックを消した場合と同様に上にあるブロックは,下へ移動する.
スコアの加算についても横でブロックを消した場合と同様に,1列消すごとに 10pt 加算される.

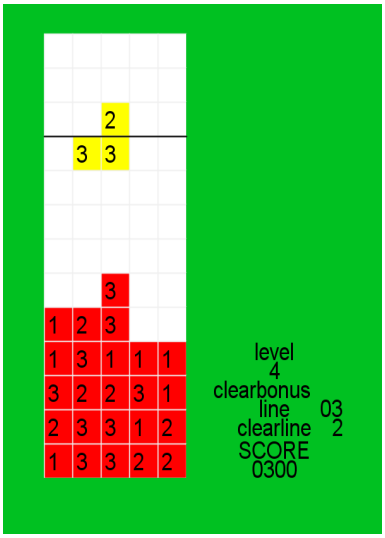


6. プレイ画面 (難易度の上昇)
ブロックを次々に消していき,スコアが 70 の倍数となると,ブロックの落下速度が難易度に応じて変化する.
難易度は Level 4 まで上昇するようになっている.

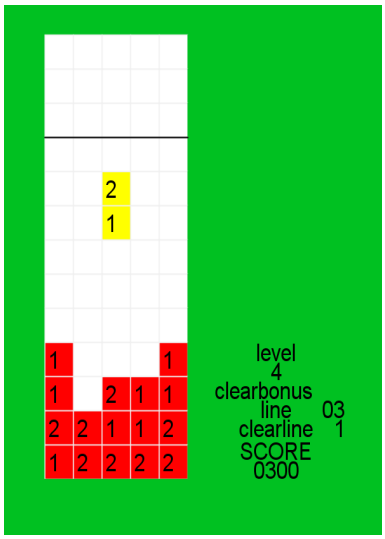
7. プレイ画面 (第一ボーナスの発生)
また,スコアが 70 の倍数となると,スペースキーを押すことで一番したの段を消すことのできるボーナスが利用可能となる.



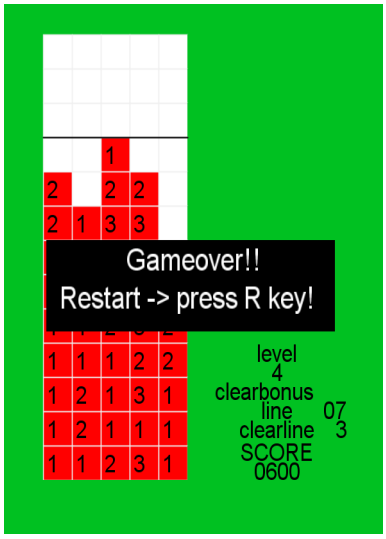
8. プレイ画面 (第一ボーナスの利用)
第一ボーナスを利用すると,一番したの段の和に関係なく消される.
そして,ボーナスを利用するとそのボーナスの利用回数が減る.
スコアに加算はされない.



9. プレイ画面 (第二ボーナスの発生)
スコアが 140 の倍数となると,1,2,3の値が割り当てられたブロックのなかで,いずれかの数字のブロックをすべて消すことのできるボーナスを使うことができるようになる.
それぞれの消したい数字のキーを押すことで利用できる.
画像では,スコアが300であるから,第二ボーナスの利用回数が2回となっている.



10. プレイ画面 (第二ボーナスの利用)
画像では,3が割り当てられたブロックを消した.
この場合も残ったブロックはそのまま下へ移動する.
そして,第一ボーナスと同様,そして,ボーナスを利用するとそのボーナスの利用回数が減り,スコアに加算はされない.
今回利用したときは,1,2,3それぞれの数字が割り当てられたブロックの数の差はあまりなかったが,よりある数のブロックの個数に偏りがあった場合にこのボーナスを利用すると,よりゲームを続けることに繋げられる.



11. ゲームオーバー
上から三段目の部分の黒いラインいブロックが到達してしまうと,ゲームオーバーとなってしまう.
r キーを押すことにより,難易度,ボーナスの利用回数,スコアをすべて初期かして,再びゲームをすることができる.

- 試用結果

　ゲームの難易度としては,ブロックの落下してくる速度といかに早くかつ素早くブロックを縦もしくは横で5個並べて合計を 10 とすることができるかという部分があるが,これに関しては,当初予定していたものが実現できているようにも思われる.また,当初の予定ではユーザーが操作しているブロックとは別に,次のブロックを表示させることを考えていた.しかし,瞬時に計算を行わせる一つの方法として,次のブロックが表示されない方がより早い計算力を必要となるように思えたので,次のブロックが表示される仕様は今回は見送ることとなった.

　また,本来ならば難易度が上がるごとに落下してくるブロックの種類が増える仕様,ブロックの値の種類を幅広くするような仕様,一つのラインを消すために必要となる値を変更する仕様の実装を考えていたが,ゲームの枠を 10 × 5 で作成したため,それらの仕様の実現にはゲームの枠の変更をしなければならないなどの意見により,これらの仕様の実装も見送りとした.

　今回作成したゲームでは,10 × 5 の枠ということもあり,ブロックに割り当てられる数字が 1 ～ 3 と 3 種類しかない.そして,これらを乱数によって生成してブロック一つ一つに割り当てている.その結果,少ない種類の数字を乱数によって生成しているため,同じ数字が何度も落下してくる,全くある数字が落下してこないなどの問題もあるように思えた.

- 今後の課題

　今後の課題としては,まずはプログラムに関して,コンストラクタやオーバーライドなどを利用して,より効率よく,さらにわかりやすいプログラムを制作できるようにしたい.今回,チームを編成して各チームでゲームを制作するという課題であったが,我々のチームでは各クラスの制作にあたり,チームメンバー内でどのような変数を利用するのかなどの細かな話し合いをあまりしなかったため,無駄な変数があるように思えたためである.

　仕様結果にも記述されているが,同じ数字が何度も落下してくる,全くある数字が落下してこないなどの問題があるので,それを改善することができればより良いゲームができると考えている.

　ゲーム内部に関しては,ゲームのスタート画面から Enter キーを押したらすぐにゲームが始まるのではなく,ブロックの落下速度の変更やボーナス利用の有無,操作しているブロックの次に現れるブロックの表示の有無などのユーザーが難易度を自分で変更できるようにするとよりユーザー好みの設定ができ,さらにゲーム性が向上していくように思えるので,そのような仕様を実装できるようにしていきたい.