課題「ゲームの改変」

C++基礎

概要

与えられたプロジェクト「JumpJump」を指示された内容に改変してください。
主な改変内容は、「ソースのリファクタリング」や「追加実装」です。

この課題を行う際に必要になること

- 課題を行うためのじゅうぶんな時間の確保
  - 自分以外が書かれたソースを読むためには思ったよりも時間がかかります
  - また、求められていることを自分以外が書いたソースに入れることも時間を費やします
- デバッグ能力
  - Visual Studio でのデバッグ機能について
    - F11 を押すとプログラムの最初から一時停止しながら実行
    - ソースコード上で F9 を押すと『ブレークポイント』が設置でき、
      そのまま実行（F5）をすることで、その行を実行するタイミングでプログラムが一時停止します
    - F10 を押すと、F11 押下後、または、F5 押下後、実行する行が次の行へ移動します
  - 処理をコメントアウトして、実行画面にどんな影響があるのか確認する
  - 数値をデバッグ表示で確認する
    - DX ライブラリでのデバッグ表示用関数
      - printfDx 関数（使い方は printf_s 関数と同じ）
- プログラミングの基礎力
  - 変数、関数の作成、基礎構文が使えることを求めています
- プログラミング用語への理解度
  - 求められている指示にはプログラミング用語が含まれています
  - 求められている指示を遂行するためにはプログラミング用語を理解している必要があります
- ソースコードの読解力
  - それぞれの変数がどのように動いているか、それぞれの処理がどのように実行されているかなど自分以外が書かれたソースのコメントを読みながらでも追うことができるのかが必要です

- 規則通りに自分のソースを書けるか
  - 変数名、関数名、定数名などは、以下のルールに沿って宣言等してください
    - ルール
      - 長くなってもいいので意味のある、または、それぞれの使用用途をあらわす単語にすること
    - 禁止事項
      - 「a」などの意味のないアルファベット一文字
      - 「speed」などのどの処理に関わるのか判断できない汎用性のある単語
      - こちらが指定する「命名規則」に沿っていないもの
    - 命名規則

変数（引数）	dog_and_cat
自作関数	dogAndCat
構造体	DogAndCat
定数（列挙体）	DOG_AND_CAT

この課題を行ったことでできるようになること

- プログラミングの基礎力の向上
- ソース読解力の向上
- デバッグ力の向上
- 自分以外が書かれたソースに対する改変能力の向上

提出について

提出締め切り

2023 年 10 月 10 日（火）23:59

提出用フォーム

https://forms.office.com/r/GchATXYxwi

提出物

- Release モードで実行をした exe ファイル
- プログラムを記載したソースファイル
  - ヘッダーファイル等を使用した場合は、ヘッダーファイルも提出物に含みます

提出方法

フォルダ名を以下の命名規則にのっとり、そのフォルダ内に提出物を入れ、ZIP 形式に圧縮をして、ギガファイル便にアップロードしてください。
ギガファイル便にアップロード後、ダウンロード URL を指定のフォームへ記載をして送信を押してください。
Teams 内にデータをアップロードするわけではないことに注意をしてください。

フォルダ名「XXNN_Name」

フォルダ名の命名規則の説明	
XX	スーパーゲームの学生は「SPG」、ゲープロの学生は「PG」
NN	学籍番号下二桁
Name	自身の名前のフルネーム（半角英字のみ使用）

例：
スーパーゲーム科、学籍番号 202304180000 の金丸のフォルダ名は以下の通りです。
SPG00_KanamaruNatsuko

※ 提出物・提出形式に誤りが多く見られます、提出する際には確認を何度もしてください

ゲームの仕様

この「Jump!Jump!!」ゲームは、現状は以下の通りです。

画面について

- 画面サイズは、幅 960 ピクセル、高さ 540 ピクセルです
- ただし、画面サイズは開発途中で変更がある可能性があるため、画面サイズの幅 960 を使用する部分では「WINDOW_W」、画面サイズの高さ 540 を使用する部分では「WINDOW_H」を使用するようにしてください

プレイヤーについて

- 画面左側にいる、青色の箱がプレイヤーです
- スペースキー押下でジャンプをし、2 段まで可能です
- 敵に当たりダメージを受けると 1 秒間点滅をしますが、当たり判定は実装されていないので A キーを押すことでダメージが与えられた状態のプレイヤーを確認することができます
- 座標は四角の中心とします
  - 左上などに変更しないように注意してください

敵（壁）について

- 画面右端から左に移動する赤い壁が敵です
- 敵の数は定数「ENEMY_MAX_NUM」で管理しています
  - この定数に代入されている数値を変更することで敵の数の増減ができます
- 移動速度は定数「ENEMY_SPEED」で管理しています
  - この定数に代入されている数値を変更することで敵の移動速度を加減できます
- プレイヤーがこの敵に当たるとプレイヤーはダメージを受けますが、当たり判定は実装されていません
- 敵の頭上にある数値は、配列変数「enemy_x」などの配列のインデックス値をあらわしているデバッグ表示です
  - 「GameMain.cpp」の 203 行目以降がその処理を行っているところです
  - 実装がすべて終わった際には「GameMain.cpp」の 203 行目以降は削除予定です
- 座標は四角の中心とします
  - 左上などに変更しないように注意してください

地面について

- 画面下部にある黒い四角は、床に見立てた地面です
- 高さは 60 ピクセルです
- 地面の Y 座標は「GROUND_Y」という定数に代入されています

操作方法

スペースキー	ジャンプ（ジャンプ中も押下可能、2 段までジャンプ可能）
A キー	ダメージ処理の発動用のデバッグキー

四角の描画について

- 四角の中心が座標になるように四角の描画は、Common.h/Common.cpp に書かれている「drawBox 関数」を使用しています
- 今後四角を描画する際は、DX ライブラリの関数「DrawBoxAA 関数」等は使用せず、「drawBox 関数」を使用してください
  - DX ライブラリの関数「DrawBoxAA 関数」等の使用は禁止です

Common.h/Common.cpp について

- ゲーム中に「共通して使用するもの」に関してはこのファイルを使用します

改変内容

求められている改変内容は次の通りです。

それぞれの項目に対しての詳細は、各項目の詳細表記を参照してください。

- 定数の作成と適用
- 構造体「Box」の宣言と適用
- プレイヤーと敵との当たり判定の実装
- バグの修正
- プレイヤーに関わる処理をすべて別ファイルへ移動

定数の作成と適用

以下の数値を定数にしてください。
なお、定数は「#define」を使わず、変数の宣言に「const」を付けて宣言して使用してください。

- プレイヤーのダメージ時間を「60」としていますが、この「60」という数値を定数化してください
  - 他のところでも「60」という数値を使用していますが、定数化するのはプレイヤーのダメージ時間のみですので、対応には注意してください
- プレイヤーのジャンプ力を定数化してください
  - 1 回目のジャンプ力は「20.0f」
  - 2 回目のジャンプ力は「25.0f」

構造体「Box」の宣言と適用

プレイヤーも敵も四角という図形を使用しており、プレイヤーも敵もそれぞれ同じ用途の変数を持っています。
バラバラに宣言された変数を構造体「Box」としてまとめ、その構造体を型としたオブジェクト（変数）を作成して、バラバラに宣言されている変数をまとめてください。

構造体「Box」にまとめる変数について

- Ｘ座標
  - 浮動小数点数型の構造体のメンバ（変数）です
  - 四角のＸ座標を扱います
- Ｙ座標
  - 浮動小数点数型の構造体のメンバ（変数）です
  - 四角のＹ座標を扱います
- 幅
  - 浮動小数点数型の構造体のメンバ（変数）です
  - 四角のサイズである幅を扱います
- 高さ
  - 浮動小数点数型の構造体のメンバ（変数）です
  - 四角のサイズである高さを扱います
- 色
  - 整数型（符号なし）型の構造体のメンバ（変数）です
  - 四角の色を扱います

構造体「Box」の宣言箇所について

構造体「Box」は、Common.h の該当箇所に宣言してください。

構造体「Box」の適用箇所について

構造体「Box」を適用する場所に該当するソースは、次の通りです。

- プレイヤー用変数の関連部分
- 敵用変数の関連部分
- drawBox 関数の引数
  - drawBox 関数の引数は、構造体「Box」を参照渡し（const 付き）にしてください

プレイヤーと敵との当たり判定の実装

プレイヤーと敵の当たり判定の実装として、当たり判定用の関数を「Hit.h/Hit.cpp」に作成してください。
作成した当たり判定用関数を GameMain.cpp で使用して、プレイヤーと敵との当たり判定を行い、もし当たっている場合、プレイヤーにはダメージ処理、敵は画面外へと移動させるようにしてください。
なお、プレイヤーがダメージ中はプレイヤーと敵との当たり判定を行わないようにしてください。

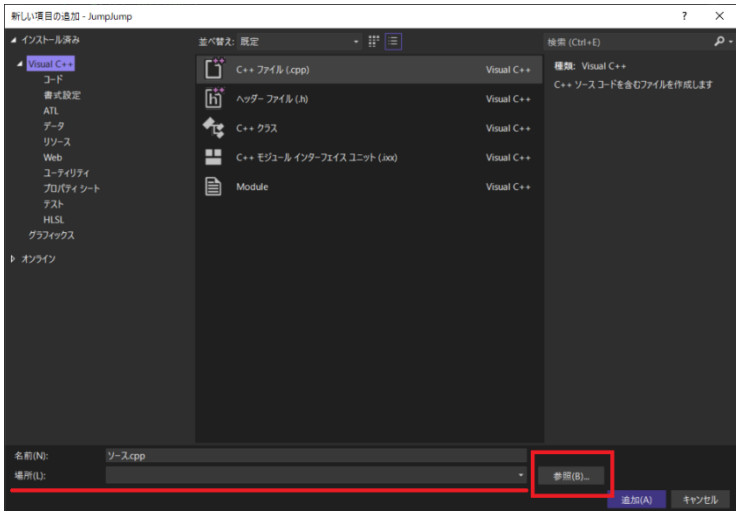
当たり判定用関数のファイルの作成について

ファイル名は次の通りです。

- ヘッダーファイル
  - Hit.h
- ソースファイル
  - Hit.cpp

ファイルを作成する際の注意点は以下の通りです。

- ファイルを追加するダイアログボックスが開いた際、作成する場所を「src フォルダ」を指定してください
  - 「参照」ボタンを押すことでファイルを作成する場所を指定できます



四角と四角との当たり判定について

関数に引数と戻り値は次の通りです。

- 引数
  - 当たり判定の対象物 1
    - 構造体「Box」を参照渡し（const 付き）
  - 当たり判定の対象物 2
    - 構造体「Box」を参照渡し（const 付き）
- 戻り値
  - 当たった場合は true、当たらなかった場合は false を返す

当たり判定の処理は、一任します。

作成した当たり判定の使用について

GameMain.cpp で使用しますので、GameMain.cpp の冒頭でインクルードしてください。（#include “Common.h”の下に挿入する）

プレイヤーと敵との当たり判定をして、もしプレイヤーと敵が当たっている場合には、次の処理を行ってください。

- プレイヤーの処理
  - ダメージ処理を行う
  - ダメージ処理詳細
    - 60 フレーム間点滅
      - 点滅中は無敵状態となり、敵との当たり判定を行わない
- 敵の処理
  - 画面外へ移動し、再度待機時間の設定、その後右から左へ移動を行う

バグの修正

敵の出現に際し、バグが発生しています。
そのバグとは「敵の出現が一定間隔になっているためにプレイヤーが敵を絶対に避けられないゲームになっている」というところです。
敵の出現を一定間隔ではなく、ランダムで、また、敵同士が重ならずに出現するように修正対応を行ってください。
なお、その際にグローバル変数を追加すること、関数を追加することは許可します。
ただし、敵の座標や待機時間の設定は、GameInit 関数内と GameUpdate 関数内の 2 か所で行っていることに注意をして実装してください。

プレイヤーに関わる処理をすべて別ファイルへ移動

プレイヤーに関係する処理、その処理に関するグローバル変数・列挙体をプレイヤー用のファイルに移動してください。
ただし、当たり判定の処理は引き続き GameMain.cpp 内で行うこととします。

プレイヤー用ファイルの作成について

ファイル名は次の通りです。

- ヘッダーファイル
  - Player.h
- ソースファイル
  - Player.cpp
    - このファイルの冒頭で「WinMain.h」をインクルードする
      - インクルードしないと DX ライブラリの関数などが使えません
    - このファイルの冒頭で「Common.h」をインクルードする
      - インクルードしないと構造体「Box」が使えません
    - このファイルの冒頭で「Player.h」をインクルードする

ファイルを作成する際の注意点は Hit.h/Hit.cpp を作成した際と同じです。

グローバル変数の移動について

グローバル変数の「プレイヤー用変数」を「Player.cpp」へ移動してください。
ただし、各グローバル変数冒頭についている内部リンケージ用の「static」は外さないでください。

プレイヤー処理用関数の作成と移動について

Player.h/Player.cpp に宣言、定義する関数は次の通りです。(Player.h では、関数プロトタイプのみ行い、関数の実装は Player.cpp で行うこと)

- playerInit 関数
  - 引数・戻り値なし
  - プレイヤーの初期化処理を行う関数です
  - GameInit 関数内の「// プレイヤー」と書かれているところと書かれているところをこの関数内に移動し、この関数は外部リンケージ化する
- playerUpdate 関数
  - 引数・戻り値なし
  - プレイヤーの更新処理を行う関数です
  - GameUpdate 関数内の「// プレイヤー」と書かれているところをこの関数内に移

動し、この関数は外部リンケージ化する
✧　ただし、ダメージ処理は別途関数を作成する

- playerRender 関数
  - 引数・戻り値なし
  - プレイヤーの描画を行う関数です
  - GameDraw 関数内の「// プレイヤー」と書かれているところをこの関数内に移動し、この関数は外部リンケージ化する
- playerSetDamage 関数
  - 引数・戻り値なし
  - プレイヤーにダメージ処理を与える関数です
  - ダメージが入った瞬間にだけ呼ばれる関数です
  - GameUpdate 関数内の「// ダメージ開始」と書かれているところをこの関数内に移動し、この関数は外部リンケージ化する
- playerDamageUpdate 関数
  - 引数・戻り値なし
  - プレイヤーのダメージ中の処理を行う関数です
  - GameUpdate 関数内の「// ダメージ中処理」と書かれているところをこの関数内に移動し、この関数は Player.cpp でしか使わないため、内部リンケージの関数とする
  - また、playerUpdate 関数の最後にこの関数を呼ぶようにする

作成した関数を GameMain.cpp で使用について

GameMain.cpp でプレイヤーの処理を使用するために、GameMain.cpp の冒頭で Player.h をインクルードしてください。(#include “Hit.h”の下に挿入する)
それぞれの処理が入っていた箇所に作成したそれぞれの関数を呼び出すようにしてください。

当たり判定でのエラーへの対処について

プレイヤー用変数を Player.cpp に移動したため、当たり判定をしているところでエラーが発生するようになってしまいました。
エラーが発生するようになった具体的な原因は以下の通りです。

- プレイヤー用変数は Player.cpp で“内部リンケージ”として宣言されているため、Player.cpp ファイル外では使用することができない

ここで発生しているエラーを「Player.cpp に宣言されたグローバル変数の static を外さず」、かつ、「extern を使用せず」に対応してください。
なお、この対応に伴い、プレイヤー処理用関数を追加することは許可されています。