講義メモ
テキスト編：「アレンジ演習：p.291 clock01.cs」から
ゲーム開発演習：背景画面のスクロール、アイテムの同時スクロール、左右移動　など

アレンジ演習：p.291 clock01.cs

・ストップウォッチにしよう
・実行すると「00:00:00」を表示してカウントを開始するようにしよう
・そのために、現在時刻ではなく、時刻を表す1000万分の1秒刻みのカウンタをDateTime構造体のTicksプロパティ(long型)で得て用いる
・起動時に現在時刻を持つオブジェクトを得たらTicksを確保しておき、最新のTicksとの差を算出する
・この差を1000万倍すると秒になるので、3600で割って時間を、60で割った結果の60の剰余で分を、60の剰余で秒を得ると良い

作成例

```
//アレンジ演習：p.291 clock01.cs
using System;
class clock01 {
    public static void Main() {
        int oldsecond = 0; //秒を比較用に保持する変数
        Console.CursorVisible = false; //カーソルを非表示に
        Console.Title = "時計"; //コンソールタイトル設定
        Console.SetWindowSize(12, 3); //コンソールの大きさ設定
        Console.BackgroundColor = ConsoleColor.Yellow; //背景色
        Console.ForegroundColor = ConsoleColor.Black; //文字色
        Console.Clear(); //変更を反映
        DateTime st = DateTime.Now; //【追加】開始時の日付時刻オブジェクト
        DateTime mt; //日付時刻オブジェクト用
        while (true) { //無限ループ
            mt = DateTime.Now; //現在日付時刻を得る
            int lap = (int)((mt.Ticks - st.Ticks) / 10000000); //【追加】経過秒数を得る
            int Hour = lap / 3600; //【追加】時間を得る
            int Minute = lap / 60 % 60; //【追加】分を得る
            int Second = lap % 60; //【追加】秒を得る
            if (Second == oldsecond) { //【変更】秒が変わっていない？
                continue; //後続処理をスキップして次へ
            } else { //秒が変わっている？
                oldsecond = Second; //【変更】新しい秒を取っておく
            }
            Console.SetCursorPosition(2, 1); //カーソルを前へ移動
            Console.Write("{0:00}:{1:00}:{2:00}",
                Hour, Minute, Second); //【変更】時分秒を各2桁で表示
            if (Console.KeyAvailable) { //何かキーが押された？}
                break; //繰返しを抜ける
            }
        }
```

```
    }
}
```

アレンジ演習：p.291 clock01.cs・続き

・ミリ秒までのストップウォッチにしよう
・実行すると「00:00:00.000」を表示してカウントを開始するようにしよう（表示幅を4文字分増やす）
・最新のTicksとの差を1万倍するとミリ秒になるので、3600000で割って時間を、60000で割った結果の60の剰余で分を、1000で割った結果の60の剰余で秒を、1000の剰余でミリ秒を得ると良い

作成例

```
//アレンジ演習：p.291 clock01.cs
using System;
class clock01 {
    public static void Main() {
        int oldmsecond = 0; //【変更】ミリ秒を比較用に保持する変数
        Console.CursorVisible = false; //カーソルを非表示に
        Console.Title = "時計"; //コンソールタイトル設定
        Console.SetWindowSize(16, 3); //【変更】コンソールの大きさ設定
        Console.BackgroundColor = ConsoleColor.Yellow; //背景色
        Console.ForegroundColor = ConsoleColor.Black; //文字色
        Console.Clear(); //変更を反映
        DateTime st = DateTime.Now; //開始時の日付時刻オブジェクト
        DateTime mt; //日付時刻オブジェクト用
        while (true) { //無限ループ
            mt = DateTime.Now; //現在日付時刻を得る
            int lap = (int)((mt.Ticks - st.Ticks) / 10000); //【変更】経過
秒数を得る
            int Hour = lap / 3600000; //【変更】時間を得る
            int Minute = lap / 60000 % 60; //【変更】分を得る
            int Second = lap / 1000 % 60; //【変更】秒を得る
            int MSecond = lap % 1000; //【追加】ミリ秒を得る
            if (MSecond == oldmsecond) { //【変更】ミリ秒が変わっていない？
                continue; //後続処理をスキップして次へ
            } else { //秒が変わっている？
                oldmsecond = MSecond; //【変更】新しいミリ秒を取っておく
            }
            Console.SetCursorPosition(2, 1); //カーソルを前へ移動
            Console.Write("{0:00}:{1:00}:{2:00}.{3:000}",
                Hour, Minute, Second, MSecond); //【変更】時分秒を各2桁で表示
            if (Console.KeyAvailable) { //何かキーが押された？}
                break; //繰返しを抜ける
            }
        }
    }
}
```

p.294 練習問題 ヒント

・プロパティによる制限なので、エラー表示は含まなくて良い
・偶数は正の整数なので、構造体に含まれるメンバの型はuintにすると良い
・偶数しか保持できないようにするには、データメンバへの直接アクセスを禁止するしかない
・よって、データメンバはprivateとしよう
・そして、このデータメンバを扱うプロパティはpublicとする
・プロパティのsetにおいて、valueをチェックし、偶数であれば代入する
・プロパティのgetは通常通り

作成例

```
//p.294 練習問題
using System;
struct MyStruct { //構造体定義
    private uint x; //構造体のデータメンバ
    public uint X { //プロパティ
        get { return x; }
        set { if (value % 2 == 0) { x = value; } }
    }
}
class struct01 {
    public static void Main() {
        MyStruct ms = new MyStruct(); //newが必要
        ms.X = 10; //構造体のプロパティで代入
        Console.WriteLine(ms.X); //構造体のプロパティを呼ぶ
        ms.X = 11; //構造体のプロパティで代入(できない)
        Console.WriteLine(ms.X); //構造体のプロパティを呼ぶ
    }
}
```

第12章 デリゲートとイベント

p.295 デリゲートとは

・メソッドへの参照を保持しておいて、これを用いてメソッドを呼び出せる仕掛け
・C/C++における「関数へのポインタ」の考え方を洗練したもの
・C#公式リファレンスでは「代理人」と和訳されていることがあるが、ニュアンスが異なる
・主に、イベントなどで用いる
・利用には宣言と生成が必要で、宣言はクラスの外で行う
・宣言書式: delegate メソッドの戻り値型 デリゲート名(メソッドの引数リスト);
・この書式でわかる通り、デリゲートで用いたいメソッドと、戻り値型、引数リストが一致している必要がある
・例: delegate bool md(int w); //このデリゲートで「bool foo(int x){…}」などが扱える
・デリゲートの生成において、扱うメソッド名を指定する
・生成書式: デリゲート名 参照変数 = new デリゲート名(メソッド名);
・例: md work = new md(foo); //fooメソッドを呼び出せるデリゲートを生成しworkとする
・デリゲート経由でメソッドを呼び出すには、参照変数をメソッドの別名にように扱える
・例: bool ans = work(12); // bool ans = foo(12);と同じ動作になる

p.297 delegate01.cs

```
//p.297 delegate01.cs
using System;
delegate void MyDelegate(); //デリゲートの宣言(戻り値無、引数無))
class delegate01 {
    public static void show() { //静的メソッド(Mainから呼出可)
        Console.WriteLine("呼ばれました");
    }
    public static void Main() {
        //直接showメソッドを呼び出す
        show();
        //デリゲートの作成
        MyDelegate md = new MyDelegate(show);
        //デリゲートを通してshowメソッドを実行
        md();
    }
}
```

p.298（別のクラスにあるインスタンスメソッドをデリゲート経由で呼び出す）

・別のクラスにあるインスタンスメソッドをデリゲート経由で呼び出すことができる
・そのクラスのインスタンスを生成して用いると良い
・生成書式: デリゲート名 参照変数 = new デリゲート名(インスタンス名.メソッド名);

作成例

```
//p.298 delegate02.cs
using System;
delegate void MyDelegate(); //デリゲートの宣言(戻り値無、引数無))
class MyClass {
    public void show() { //インスタンスメソッド
        Console.WriteLine("呼ばれました");
    }
}
class delegate02 {
    public static void Main() {
        MyClass mc = new MyClass(); //インスタンスを生成
        mc.show(); //インスタンスで直接呼出す
        MyDelegate m = new MyDelegate(mc.show); //デリゲートを生成
        m(); //デリゲート経由で呼出す(インスタンス名は不要)
    }
}
```

テキスト編次回予告:p.298「別のクラスにある静的メソッドをデリゲート経由で呼び出す」から

ゲーム開発演習:背景画面のスクロール、アイテムの同時スクロール、左右移動 など

テーマ25 背景画面のスクロール（再掲載＆変更）

・GDI+の座標系では画面に表示されない範囲外の座標を指定しても良い
・そのため、画像や図形の描画開始位置を範囲外にして、範囲内にある部分のみを表示してOK
・この仕組みを活用して背景画面のスクロールをすることができる
・以下は縦スクロールの場合だが、横スクロールも同様。
・背景画像を2枚用意し、1枚目の描画開始位置を画面左上(0,0)から順次下に変更していく
・すると、1枚目が見かけ下に移動して、上が開くので、そこに2枚目を描画すればよい
・そして完全に下がり切ったら、元の(0,0)に戻せば良い

演習21 背景画面のスクロール

・背景画像backiを縦に下方向へスクロールするようにしよう
・背景画像backiは上下がつながるデザインなので1枚の画像を2つ並べて表示すると良い
・1枚目の描画開始Y座標を保持する変数を0で初期化し、タイマーで1画面分までインクリメントすれば良い
・そして、画像の高さ分まで進んだら0に戻せばよい(画像の高さで割った余りにする)
・タイマーのインターバルを10ミリ秒にしよう
・併せて、同心円と矩形は削除し、スコアの加算を中止しよう

作成例

```
//演習21 背景画面のスクロール
using System; //汎用的に利用
using System.Windows.Forms; //フォームアプリケーションに必須
using System.Drawing; //Size、Image用
class Program : Form { //Formクラスの派生クラス
    int gamemode = 0; //モード(0:タイトル画面,1:プレイ画面,9:終了画面)
    int score = 0; //スコア
    Image backi = Image.FromFile("backb.bmp"); //背景画像を読込む
    Image numx = Image.FromFile("numx.bmp"); //アイテム画像を読込む
    Pen pen1 = new Pen(Color.Red, 2); //赤色太さ2のペン
    Brush brush1 = new SolidBrush(Color.FromArgb(63, 255, 0, 0)); //透明赤いブラシ
    Font font1 = new Font("メイリオ", 20, FontStyle.Bold); //フォントを生成
    Font fontt = new Font("メイリオ", 80, FontStyle.Bold); //フォントを生成
    Font fontm = new Font("メイリオ", 25, FontStyle.Bold); //フォントを生成
    Brush brushs = new SolidBrush(Color.Yellow); //黄色のブラシ
    Timer timer = new Timer(); //タイマーの生成
    int backy = 0; //【追加】1枚目の背景描画開始Y座標
    protected override void OnPaint(PaintEventArgs e) { //描画処理のオーバライド
        base.OnPaint(e); //基本クラスの描画処理を呼ぶ
        e.Graphics.DrawImage(backi, 0, backy); //【変更】背景画像を描画
        e.Graphics.DrawImage(backi, 0, backy - backi.Height); //【追加】背景画像を描画
        if (gamemode == 0) { //スタート画面？
            e.Graphics.DrawString("GAME1", fontt, brushs, 100, 150); //タイトル表示
```

```
            e.Graphics.DrawString("Hit Enter Key", fontm, brushs, 200,
300); //メッセージ表示
        } else if (gamemode == 1) { //プレイ画面？
            string s = String.Format("SCORE:{0:000,000}", score); //スコア
文字列を作る
            e.Graphics.DrawString(s, font1, brushs, 400, 10); //スコア表示
            int x = backi.Width / 2, y = backi.Height / 2; //中心座標を得
る
            e.Graphics.DrawImage(numx, x - numx.Width / 2, y -
numx.Height / 2); //アイテム画像を描画
            //e.Graphics.FillRectangle(brush1, 78, 411, 485, 64); //【削
除】矩形を塗りつぶす
            //e.Graphics.DrawRectangle(pen1, 78, 411, 485, 64); //【削除】
矩形を描く
            //pen1.Color = Color.Yellow; //【削除】ペンを黄色にする
            //pen1.Width = 10; //【削除】ペン太さを10にする
            //for (int i = 1; i <= 4; i++) { //【削除】4回繰返す
            //    e.Graphics.DrawEllipse(pen1, x - 15 * i, y - 15 * i, 30
* i, 30 * i); //【削除】円を描く
            //}
        }
    }
    void OnKeyDown(object o, KeyEventArgs e) { //キー入力時処理
        if (e.KeyCode.ToString() == "Escape") { //Escキーが押されていたら
            Close(); //フォーム終了
        }
        //タイトル画面でEnterキーが押されていたら
        if (gamemode == 0 && e.KeyCode.ToString() == "Return") {
            gamemode = 1; //プレイ動画に遷移
            timer.Start(); //タイマー開始
        }
        Invalidate(); //画面再描画を依頼
    }
    void Play(object o, EventArgs e) { //タイマーイベント処理
        //score++; //【削除】スコアカウントアップ
        backy = (backy + 1) % backi.Height; //【追加】1枚目の背景描画開始Y座
標を下げる
        Invalidate(); //画面再描画を依頼
    }
    Program() { //コンストラクタ
        DoubleBuffered = true; //ダブルバッファリングを有効化
        KeyDown += new KeyEventHandler(OnKeyDown); //キー入力イベント登録
        timer.Tick += new EventHandler(Play); //タイマーイベント登録
        timer.Interval = 10; //【変更】タイマーインターバル(ミリ秒)
    }
    public static void Main() {
        Program f = new Program(); //自分のオブジェクトを生成
        f.Size = new Size(660, 520); //フォームのサイズを設定
        f.Text = "Game"; //フォーム名を設定
```

```
        f.ControlBox = false; //コントロールボックスを非表示に
        f.FormBorderStyle = FormBorderStyle.Fixed3D; //サイズ変更を抑止
        Application.Run(f); //フォームを現出
    }
}
```

テーマ26 アイテムの同時スクロール

・背景画像の上においたアイテム画像を背景と同速で動かせば同時スクロールが可能
・ただし、画面外へのはみだしや、画面上部からの再出現が必要であれば、別途管理すること

演習22 アイテムの同時スクロール

・アイテムnumxを背景と同速で縦に下方向へスクロールするようにしよう
・完全に画面外に出たら描画とスクロールは中止しよう

作成例

```
//演習22 アイテムの同時スクロール
using System; //汎用的に利用
using System.Windows.Forms; //フォームアプリケーションに必須
using System.Drawing; //Size、Image用
class Program : Form { //Formクラスの派生クラス
    int gamemode = 0; //モード(0:タイトル画面,1:プレイ画面,9:終了画面)
    int score = 0; //スコア
    Image backi = Image.FromFile("backb.bmp"); //背景画像を読込む
    Image numx = Image.FromFile("numx.bmp"); //アイテム画像を読込む
    Pen pen1 = new Pen(Color.Red, 2); //赤色太さ2のペン
    Brush brush1 = new SolidBrush(Color.FromArgb(63, 255, 0, 0)); //透明
赤いブラシ
    Font font1 = new Font("メイリオ", 20, FontStyle.Bold); //フォントを生成
    Font fontt = new Font("メイリオ", 80, FontStyle.Bold); //フォントを生成
    Font fontm = new Font("メイリオ", 25, FontStyle.Bold); //フォントを生成
    Brush brushs = new SolidBrush(Color.Yellow); //黄色のブラシ
    Timer timer = new Timer(); //タイマーの生成
    int backy = 0; //1枚目の背景描画開始Y座標
    int numxy = 0; //【追加】アイテムの描画Y座標の増分
    protected override void OnPaint(PaintEventArgs e) { //描画処理のオーバラ
イド
        base.OnPaint(e); //基本クラスの描画処理を呼ぶ
        e.Graphics.DrawImage(backi, 0, backy); //背景画像を描画
        e.Graphics.DrawImage(backi, 0, backy - backi.Height); //背景画像を
描画
        if (gamemode == 0) { //スタート画面？
            e.Graphics.DrawString("GAME1", fontt, brushs, 100, 150); //タ
イトル表示
            e.Graphics.DrawString("Hit Enter Key", fontm, brushs, 200,
300); //メッセージ表示
        } else if (gamemode == 1) { //プレイ画面？
```

```
            string s = String.Format("SCORE:{0:000,000}", score); //スコア
文字列を作る
            e.Graphics.DrawString(s, font1, brushs, 400, 10); //スコア表示
            int x = backi.Width / 2 - numx.Width / 2; //【変更】
            int y = backi.Height / 2 - numx.Height / 2 + numxy; //【変更】
中心座標を得る
            if (y < backi.Height) { //【追加】画面内なら
                e.Graphics.DrawImage(numx, x, y); //【変更】アイテム画像を描画
            }
        }
    }
    void OnKeyDown(object o, KeyEventArgs e) { //キー入力時処理
        if (e.KeyCode.ToString() == "Escape") { //Escキーが押されていたら
            Close(); //フォーム終了
        }
        //タイトル画面でEnterキーが押されていたら
        if (gamemode == 0 && e.KeyCode.ToString() == "Return") {
            gamemode = 1; //プレイ動画に遷移
            timer.Start(); //タイマー開始
        }
        Invalidate(); //画面再描画を依頼
    }
    void Play(object o, EventArgs e) { //タイマーイベント処理
        backy = (backy + 1) % backi.Height; //1枚目の背景描画開始Y座標を下
げる
        if (numxy - numx.Height < backi.Height / 2) { //【以下追加】背景高さ
の半分まで
            numxy++; //アイテムの描画Y座標の増分加算
        }
        Invalidate(); //画面再描画を依頼
    }
    Program() { //コンストラクタ
        DoubleBuffered = true; //ダブルバッファリングを有効化
        KeyDown += new KeyEventHandler(OnKeyDown); //キー入力イベント登録
        timer.Tick += new EventHandler(Play); //タイマーイベント登録
        timer.Interval = 10; //【変更】タイマーインターバル(ミリ秒)
    }
    public static void Main() {
        Program f = new Program(); //自分のオブジェクトを生成
        f.Size = new Size(660, 520); //フォームのサイズを設定
        f.Text = "Game"; //フォーム名を設定
        f.ControlBox = false; //コントロールボックスを非表示に
        f.FormBorderStyle = FormBorderStyle.Fixed3D; //サイズ変更を抑止
        Application.Run(f); //フォームを現出
    }
}
```

テーマ27 キーボードの状態を得る

・キー入力イベントを用いる手法は「キーを押している間、〇〇する」には向かない
・代わりにタイマーイベントを用いてキーボードの状態を得る処理を呼び出してもらうと良い
・これを実現するには、WindowsAPIを提供するDLL（動的リンクライブラリ）の一つである「user32.dll」を直接インポートする
・インポートの書式：[System.Runtime.InteropServices.DllImport("user32.dll")]
※セミコロン不要
・すると、これに含まれるGetKeyStateメソッドを外部定義指定により利用可能になる
・外部定義指定の書式：private static extern short GetKeyState(int nVirtKey);
・これでGetKeyStateメソッドに引数としてKey列挙子をint型にキャストして与えると、そのキーが押されていれば負の数が返される
・インポートと外部定義指定はクラス定義の先頭で行うこと

演習23 上矢印キーが押されていたらスコアアップ

・上矢印キーのKey列挙子はKeys.Up
・これを用いて、Playメソッド内で、上矢印キーが押されているかチェックし、押されていたらスコアをインクリメントしよう

作成例

```
//演習23 上矢印キーが押されていたらスコアアップ
using System; //汎用的に利用
using System.Windows.Forms; //フォームアプリケーションに必須
using System.Drawing; //Size、Image用
class Program : Form { //Formクラスの派生クラス
    [System.Runtime.InteropServices.DllImport("user32.dll")] //【追加】DLLインポート
    private static extern short GetKeyState(int nVirtKey); //【追加】外部定義指定
    int gamemode = 0; //モード(0:タイトル画面,1:プレイ画面,9:終了画面)
    int score = 0; //スコア
    Image backi = Image.FromFile("backb.bmp"); //背景画像を読込む
    Image numx = Image.FromFile("numx.bmp"); //アイテム画像を読込む
    Pen pen1 = new Pen(Color.Red, 2); //赤色太さ2のペン
    Brush brush1 = new SolidBrush(Color.FromArgb(63, 255, 0, 0)); //透明赤いブラシ
    Font font1 = new Font("メイリオ", 20, FontStyle.Bold); //フォントを生成
    Font fontt = new Font("メイリオ", 80, FontStyle.Bold); //フォントを生成
    Font fontm = new Font("メイリオ", 25, FontStyle.Bold); //フォントを生成
    Brush brushs = new SolidBrush(Color.Yellow); //黄色のブラシ
    Timer timer = new Timer(); //タイマーの生成
    int backy = 0; //1枚目の背景描画開始Y座標
    int numxy = 0; //アイテムの描画Y座標の増分
    protected override void OnPaint(PaintEventArgs e) { //描画処理のオーバライド
        base.OnPaint(e); //基本クラスの描画処理を呼ぶ
        e.Graphics.DrawImage(backi, 0, backy); //背景画像を描画
        e.Graphics.DrawImage(backi, 0, backy - backi.Height); //背景画像を
```

```
描画
        if (gamemode == 0) { //スタート画面？
            e.Graphics.DrawString("GAME1", fontt, brushs, 100, 150); //タ
イトル表示
            e.Graphics.DrawString("Hit Enter Key", fontm, brushs, 200,
300); //メッセージ表示
        } else if (gamemode == 1) { //プレイ画面？
            string s = String.Format("SCORE:{0:000,000}", score); //スコア
文字列を作る
            e.Graphics.DrawString(s, font1, brushs, 400, 10); //スコア表示
            int x = backi.Width / 2 - numx.Width / 2;
            int y = backi.Height / 2 - numx.Height / 2 + numxy; //中心座
標を得る
            if (y < backi.Height) { //画面内なら
                e.Graphics.DrawImage(numx, x, y); //アイテム画像を描画
            }
        }
    }
    void OnKeyDown(object o, KeyEventArgs e) { //キー入力時処理
        if (e.KeyCode.ToString() == "Escape") { //Escキーが押されていたら
            Close(); //フォーム終了
        }
        //タイトル画面でEnterキーが押されていたら
        if (gamemode == 0 && e.KeyCode.ToString() == "Return") {
            gamemode = 1; //プレイ動画に遷移
            timer.Start(); //タイマー開始
        }
        Invalidate(); //画面再描画を依頼
    }
    void Play(object o, EventArgs e) { //タイマーイベント処理
        backy = (backy + 1) % backi.Height; //1枚目の背景描画開始Y座標を下
げる
        if (numxy - numx.Height < backi.Height / 2) { //背景高さの半分まで
            numxy++; //アイテムの描画Y座標の増分加算
        }
        if (GetKeyState((int)Keys.Up) < 0) { //【以下追加】↑キーが押されてい
る？
            score++; //スコアアップ
        }
        Invalidate(); //画面再描画を依頼
    }
    Program() { //コンストラクタ
        DoubleBuffered = true; //ダブルバッファリングを有効化
        KeyDown += new KeyEventHandler(OnKeyDown); //キー入力イベント登録
        timer.Tick += new EventHandler(Play); //タイマーイベント登録
        timer.Interval = 10; //タイマーインターバル(ミリ秒)
    }
    public static void Main() {
        Program f = new Program(); //自分のオブジェクトを生成
```

```
        f.Size = new Size(660, 520); //フォームのサイズを設定
        f.Text = "Game"; //フォーム名を設定
        f.ControlBox = false; //コントロールボックスを非表示に
        f.FormBorderStyle = FormBorderStyle.Fixed3D; //サイズ変更を抑止
        Application.Run(f); //フォームを現出
    }
}
```

提出：演習23　上矢印キーが押されていたらスコアアップ

ゲーム開発演習次回予告：自機の左右移動、自機の画像変更、自弾の発射など