講義メモ
テキスト編：p.288「DateTime構造体を使ってみる」から
ゲーム開発演習：画面遷移、タイマー処理、背景画面スクロール

p.288 DateTime構造体を使ってみる

・DateTime構造体：C#が提供する機能で、構造体として利用可能
・C/C++/Javaなどにおける日付時刻機能に近いが、最も洗練されていて使いやすい
・年月日時分秒ミリ秒を返すint型プロパティ：
Year、Month、Day、Hour、Minute、Second、Millsecond。
・曜日をDayOfWeek列挙型で返すDayOfWeekプロパティもある
　これは日曜日を0とする英語曜日名の列挙子でできており、表示すると英語曜日名になる
・１年の通算日（1月1日を1日目とする日数）をint型で返すのがDayOfYearプロパティ
・現在日付時刻を持つDateTime構造体オブジェクトを返すのがNow静的プロパティ
　利用例：DateTime dt = DateTime.Now;
・DateTime構造体オブジェクトに含まれる年月日のみを（時分秒をゼロにした値）をDateTime構造体型で得るのがDateプロパティ
・このDateTime構造体オブジェクトを表示すると「YYYY/MM/DD 0:00:00」形式となる。
・なお、ToShortDateString()メソッドに渡すと「YYYY/MM/DD」形式となる。
・DateTime構造体には多数のコンストラクタがあるが、年月日のみを指定する DateTime(int, int, int)や、年月日時分秒を指定する DateTime(int, int, int, int, int, int)が便利。

p.290 datetime01.cs

```
//p.290 datetime01.cs
using System;
class datetime01 {
    public static void Main() {
        DateTime dt = DateTime.Now; //現在時刻を得る
        Console.WriteLine("今日は{0}年{1}月{2}日({3})です",
            dt.Year, dt.Month, dt.Day, dt.DayOfWeek //プロパティを利用
            );
        Console.WriteLine("現在{0}時{1}分{2}秒{3}ミリセコンドです",
            dt.Hour, dt.Minute, dt.Second, dt.Millisecond); //プロパティを利用
        Console.WriteLine("dt.Date = {0}", dt.Date); //YYYY/MM/DD 0:00:00形式で表示
        Console.WriteLine("短い日付形式 = {0}", dt.ToShortDateString()); //YYYY/MM/DD 0:00:00形式で表示
    }
}
```

アレンジ演習：p.290 datetime01.cs

・曜日を日本語の曜日文字にしよう
・ヒント
　・曜日名の配列を用いると良い string[] dow = {"日","月","火","水","木","金","土"};

・DayOfWeek構造体型からint型にキャストすると、日曜日から何日目かを示す整数になる
・よってこれをdow配列の添字にすると、日本語の曜日文字が得られる

作成例

```
//アレンジ演習:p.290 datetime01.cs
using System;
class datetime01 {
    public static void Main() {
        string[] dow = { "日", "月", "火", "水", "木", "金", "土" }; //【追
加】
        DateTime dt = DateTime.Now; //現在時刻を得る
        Console.WriteLine("今日は{0}年{1}月{2}日({3})です",
            dt.Year, dt.Month, dt.Day, dow[(int)dt.DayOfWeek] //【変更】プロ
パティを利用
            );
        Console.WriteLine("現在{0}時{1}分{2}秒{3}ミリセコンドです",
            dt.Hour, dt.Minute, dt.Second, dt.Millisecond); //プロパティを利
用
        Console.WriteLine("dt.Date = {0}", dt.Date); //YYYY/MM/DD 0:00:00
形式で表示
        Console.WriteLine("短い日付形式 = {0}", dt.ToShortDateString());
//YYYY/MM/DD 0:00:00形式で表示
    }
}
```

アレンジ演習:p.290 datetime01.cs 続き

・コンソールから年、月、日を入力すると、曜日を表示する機能を追加しよう
・ヒント:年、月、日をDateTime構造体の、DateTime(int,int,int)コンストラクタに渡して、日付時刻オブジェクトを生成させて、用いると良い

作成例

```
//アレンジ演習:p.290 datetime01.cs
using System;
class datetime01 {
    public static void Main() {
        string[] dow = { "日", "月", "火", "水", "木", "金", "土" }; //【追
加】
        DateTime dt = DateTime.Now; //現在時刻を得る
        Console.WriteLine("今日は{0}年{1}月{2}日({3})です",
            dt.Year, dt.Month, dt.Day, dow[(int)dt.DayOfWeek] //【変更】プロ
パティを利用
            );
        Console.WriteLine("現在{0}時{1}分{2}秒{3}ミリセコンドです",
            dt.Hour, dt.Minute, dt.Second, dt.Millisecond); //プロパティを利
用
        Console.WriteLine("dt.Date = {0}", dt.Date); //YYYY/MM/DD 0:00:00
```

形式で表示
```
        Console.WriteLine("短い日付形式 = {0}", dt.ToShortDateString());
//YYYY/MM/DD 0:00:00形式で表示
        Console.Write("年:"); int y = int.Parse(Console.ReadLine()); //【
以下追加】
        Console.Write("月:"); int m = int.Parse(Console.ReadLine());
        Console.Write("日:"); int d = int.Parse(Console.ReadLine());
        Console.WriteLine(dow[(int)(new DateTime(y, m, d)).DayOfWeek] + "
曜");
    }
}
```

p.291（Consoleクラスの静的プロパティとメソッド）

・Consoleクラスの静的プロパティを用いると、表示コンソールのタイトル、文字色＆背景色、カーソルの有無などを操作できる。また、キー状態を得ることもできる。
 ・bool CursorVisible:trueにするとカーソルを表示、falseにすると非表示（動きのあるアプリケーション向き）
 ・string Title:タイトル文字列を設定
 ・ConsoleColor BackgroundColor:背景色をConsoleColor列挙子で設定
 ・ConsoleColor ForegroundColor:文字色をConsoleColor列挙子で設定
 ・bool KeyAvailable:キーが押されていたらtrueを、でなければfalseを返す
・Consoleクラスの静的メソッドを用いると、表示コンソールの大きさの変更、カーソルの移動やクリア動作などを実行できる
 ・void SetWindowSize(横幅文字数, 高さ文字数):表示コンソールの大きさを変更
 ・void SetCursorPosition(横位置, 縦位置):カーソルの位置を変更
 ・void Clear():コンソールを再描画し変更を反映する

p.291 clock01.cs

```
//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 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
            if (mt.Second == oldsecond) { //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 = mt.Second; //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}",
                mt.Hour, mt.Minute, mt.Second); //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
            }
        }
    }
}
```

テキスト編次回予告:「アレンジ演習:p.291 clock01.cs」から

ゲーム開発演習:画面遷移、タイマー処理、背景画面スクロール

提出フォロー:演習17 タイトル画面

・クラス変数gamemodeを0で初期化する(0:タイトル画面、1:プレイ画面、9:終了画面)
・gamemodeが0であれば、背景画像、タイトル、メッセージ「Hit Enter to Start」を表示
・gamemodeが1であれば、背景画像、アイテム、矩形を表示
・まず、タイトル画面を表示するところまで作成しよう
  例:ゲーム名「GAME1」フォント"メイリオ",サイズ80,Bold,Yellow,X=100,Y=150
  例:メッセージ「Hit Enter Key」フォント"メイリオ",サイズ25,Bold,Yellow,X=200,Y=300

作成例

```
//演習17 タイトル画面
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; //【追加】モード(0:タイトル画面,1:プレイ画面,9:終了画面)
    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); //【追加】フォントを生
成
    Font fontm = new Font("メイリオ", 25, FontStyle.Bold); //【追加】フォントを生
成
    Brush brushs = new SolidBrush(Color.Yellow); //黄色のブラシ
    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, 0); //背景画像を描画
        if (gamemode == 0) { //【以下追加】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) { //【追加】プレイ画面？
            e.Graphics.DrawString("SCORE:000,000", 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); //矩形を
塗りつぶす
            e.Graphics.DrawRectangle(pen1, 78, 411, 485, 64); //矩形を描く
            pen1.Color = Color.Yellow; //ペンを黄色にする
            pen1.Width = 10; //ペン太さを10にする
            for (int i = 1; i <= 4; i++) { //4回繰返す
                e.Graphics.DrawEllipse(pen1, x - 15 * i, y - 15 * i, 30 *
i, 30 * i); //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
        }
    }
    Program() { //コンストラクタ
        KeyDown += new KeyEventHandler(OnKeyDown); //キー入力イベント登録
    }
    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21 画面再描画（再掲載）

・プログラム側で起動後にonPaintにおいて画面の描画を変えた場合、そのままでは実画面には反映しない
・これには、システムに対する画面再描画の依頼が必要で、Windows.Forms.Controlクラスの Invalidate()メソッドを呼べば良い
・なお、onPaintメソッドのプログラム側からの呼び出しは出来ず、システムに対する画面再描画の依頼により、システムから呼ぶ必要がある

演習18 画面遷移

・gamemodeが0の時に、Enterキーが押されたらgamemodeを1にすることでタイトル画面からプレイ画面への画面遷移をしよう
・EnterキーのKeyCodeは"Return"
・Invalidate()メソッドを呼んで画面再描画を依頼しよう

作成例

```
//演習18 画面遷移
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)
    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
    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, 0); //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？
            e.Graphics.DrawString("SCORE:000,000", 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); //矩形を塗りつぶす
            e.Graphics.DrawRectangle(pen1, 78, 411, 485, 64); //矩形を描く
            pen1.Color = Color.Yellow; //ペンを黄色にする
            pen1.Width = 10; //ペン太さを10にする
            for (int i = 1; i <= 4; i++) { //4回繰返す
                e.Graphics.DrawEllipse(pen1, x - 15 * i, y - 15 * i, 30 * i, 30 * i); //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
        }
        //【以下追加】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
        }
        Invalidate(); //【追加】画面再描画を依頼
    }
    Program() { //コンストラクタ
        KeyDown += new KeyEventHandler(OnKeyDown); //キー入力イベント登録
    }
    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22 タイマー処理

・画面を自動的に変更したり、一定時間おきに何かを行いたい場合、タイマーを用いることができる
・タイマーは、System.Windows.Forms.Timerクラスのデフォルトコンストラクタで生成できる
例: `Timer timer = new Timer();`
・このインスタンスの持つTickイベント（13章で説明）に、OnKeyDownメソッドと同様に、タイマーに呼び出して欲しいメソッドを登録する。
・登録にはEventHandler(メソッド名)を「+=」する
例: `timer.Tick += new EventHandler(タイマーに呼び出して欲しいメソッド名);`
・そして、インスタンスプロパティIntervalに、動作間隔をミリ秒で指定する
例: `timer.Interval = 20;`
・最後に、インスタンスメソッドStart()を呼ぶことでタイマーが動作する
・タイマーに呼び出して欲しいメソッドの引数は「object, EventArgs」戻り値型はvoidとすること

テーマ23 数値の形式指定文字列化

・Stringクラスの静的メソッドFormatに、書式指定文字列と数字を渡して書式指定結果の文字列を得ることができる
・書式: `string 結果文字列 = String.Format("書式指定文字列", 値, …);`
・例:

```
 string str = String.Format("x = {0}, y = {1]", 10, 20); //strは"x = 10, y = 20"となる
```

演習19 タイマーによるスコアアップ

・スコア用のint型インスタンス変数scoreを0で初期化しておく

・スコア表示を個の変数の値を表示するように変更する
例: string str = String.Format("SCORE:{0:000,000}", score);
・タイマーから呼び出されるメソッドplayで、scoreをカウントアップする
・なお、スコアを書き換えたら、画面再描画を依頼すること
・タイマーのインターバルは500ミリ秒程度にすると良い
・画面のちらつきは次の演習で対処する

作成例

```
//演習19 タイマーによるスコアアップ
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
    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; //【追加】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(); //【追加】タイマーの生成
    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, 0); //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); //【追
加】スコア文字列を作る
            e.Graphics.DrawString(s, font1, brushs, 400, 10); //【変更】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); //矩形を
塗りつぶす
            e.Graphics.DrawRectangle(pen1, 78, 411, 485, 64); //矩形を描く
            pen1.Color = Color.Yellow; //ペンを黄色にする
            pen1.Width = 10; //ペン太さを10にする
```

```
            for (int i = 1; i <= 4; i++) { //4回繰返す
                e.Graphics.DrawEllipse(pen1, x - 15 * i, y - 15 * i, 30 *
i, 30 * i); //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(); //【追加】タイマー開始
        }
        Invalidate(); //画面再描画を依頼
    }
    void Play(object o, EventArgs e) { //【以下追加】タイマーイベント処理
        score++; //スコアカウントアップ
        Invalidate(); //画面再描画を依頼
    }
    Program() { //コンストラクタ
        KeyDown += new KeyEventHandler(OnKeyDown); //キー入力イベント登録
        timer.Tick += new EventHandler(Play); //【追加】タイマーイベント登録
        timer.Interval = 500; //【追加】タイマーインターバルを500ミリ秒に
    }
    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24 ダブルバッファリング

・画面の動きがあるアプリケーションでは、画面の書き換えタイミングのずれによるチラつきが起こる
・これを防ぐにはVRAM(画像メモリ)またはそれに対応するメモリを２重化し、描画し終えた画面を表示画面に高速一括転送すると良い
・これをダブルバッファリングという
・C#のGDI+では、プロパティの設定のみでダブルバッファリングを有効化できる
・しかも、.NETフレームワーク4.8以降ではSystem.Windows.FormsクラスのDoubleBufferedプロパティをtrueにするのみでOKになっている
・なお、これ以前の.NETフレームワークを用いる場合は下記を実行すると良い:

```
SetStyle(ControlStyles.DoubleBuffer |
ControlStyles.UserPaint |
ControlStyles.AllPaintingInWmPaint, true);
```

演習20 ダブルバッファリング

・コンストラクタでダブルバッファリングを有効化してチラつきがないことを確認しよう

作成例

```
//演習20 ダブルバッファリング
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
    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, 0); //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); //矩形を
塗りつぶす
            e.Graphics.DrawRectangle(pen1, 78, 411, 485, 64); //矩形を描く
            pen1.Color = Color.Yellow; //ペンを黄色にする
            pen1.Width = 10; //ペン太さを10にする
            for (int i = 1; i <= 4; i++) { //4回繰返す
                e.Graphics.DrawEllipse(pen1, x - 15 * i, y - 15 * i, 30 *
i, 30 * i); //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++; //スコアカウントアップ
        Invalidate(); //画面再描画を依頼
    }
    Program() { //コンストラクタ
        DoubleBuffered = true; //【追加】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 = 500; //タイマーインターバルを500ミリ秒に
    }
    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25 背景画面のスクロール

・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
・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)から順次上に変更していく
・すると、1枚目が見かけ上に移動して、下が開くので、そこから2枚目を描画すればよい
・そして完全に上がり切ったら、元の(0,0)に戻せば良い

提出：演習20 ダブルバッファリング

ゲーム開発演習次回予告：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