講義メモ
テキスト編：p.315「超簡単１桁加算器（Char構造体と静的メソッド、ConsoleKeyInfo構造体とプロパティ）」から
ゲーム開発演習：自弾の発射と上移動、自弾の複数化、敵機の出現

p.315 超簡単１桁加算器（Char構造体と静的メソッド）

・C#のデータ型(p.41)は、内部的には.NET型(p.42)で表現されている
・この.NET型は、構造体として実装されている。
・よって「数値型のデータ型」.Parse(文字列)メソッド(p.45)は、.NET型の構造体のメソッドになっている
　例： int.Parse()は、Int32構造体のParse()メソッド
・char型の.NET型であるChar構造体には、文字を扱うのに便利な下記のような静的メソッドが含まれている
・public static bool IsDigit(char 文字)メソッド：文字が10進数の数字かどうかを返す
・public static double GetNumericValue(char 文字)メソッド：文字を10進数の実数に変換して返す（変換できない場合は-1を返す）
※ double.Parse(文字列)と似た動作なので、double.Parse("" + 文字)と同じ動作になる。
　しかし、double.Parseは変換できない場合は例外を投げる

p.315 超簡単１桁加算器（ConsoleKeyInfo構造体とプロパティ）

・p.293のbool KeyAvailableプロパティは、Consoleクラスの静的プロパティで、キー入力があったかどうかを返す
・これを受けて、押されたキーの情報を得られるのが、ConsoleKeyInfo構造体。
・この構造体オブジェクトを生成して押されたキーの情報を格納するのが、Consoleクラスの静的メソッドであるReadKey
・public static ConsoleKeyInfo ReadKey(bool 表示){…}
・表示がtrueの場合、押されたキーの文字は表示せず、falseだと表示する
　例： ConsoleKeyInfo cki = Console.ReadKey(true); //押されたキーを得る
・ConsoleKeyInfo構造体にあるpublic char KeyCharプロパティを用いると、押されたキーの文字が得られる

p.317 event02.cs

```
//p.317 event02.cs
using System;
delegate void Handler(char ch); //デリゲートの宣言(引数有、戻り値型無し)
class EventClass { //イベント発生を担うクラス
    public event Handler KeyHit; //イベントフィールド
    public void OnKeyHit(char ch) { //イベントを発生させるメソッド
        if (KeyHit != null) //イベントフィールドがヌルでなければ
            KeyHit(ch); //デリゲート経由で呼び出す
    }
}
class Show { //イベントで呼び出されるメソッドを持つクラス
    int sum = 0;
    public void keyshow(char ch) { //イベントで呼び出されるメソッド
        if (Char.IsDigit(ch)) { //10進数字か？
            int a = (int)char.GetNumericValue(ch); //実数に変換しintにキャスト
            sum += a; //合計に足し込む
            Console.WriteLine("+ {0}", a); //入力値を表示
            Console.WriteLine("= {0}", sum); //合計を表示
        } else if (ch == 'c') {
            sum = 0; //合計をゼロクリアする
            Console.WriteLine("合計がクリアされました");
```

```
        } else { //10進数字でもクリアでもなければ
            return; //何もしないで戻る
        }
    }
}
class event02 {
    public static void Main() {
        ConsoleKeyInfo cki; //キー情報構造体
        EventClass ec = new EventClass();  //イベント発生を担うクラスのインスタ
ンス生成
        Show s = new Show();//イベントで呼び出されるメソッドを持つクラスのインス
タンス生成
        ec.KeyHit += new Handler(s.keyshow); //デリゲートにメソッドを登録。
        while (true) { //無限ループ
            if (Console.KeyAvailable) { //何かキーが押されている？
                cki = Console.ReadKey(true); //そのキーを得る
                if (cki.KeyChar == 'x') { //xキーならば
                    break; //抜ける＝プログラム終了
                } else {
                    ec.OnKeyHit(cki.KeyChar); //キー情報を渡してイベントを発生さ
せる
                }
            }
        }
    }
}
```

アレンジ演習：p.317 event02.cs

・テキストp.319の通り、ラムダ式に書き換えよう

作成例

```
//アレンジ演習：p.317 event02.cs
using System;
delegate void Handler(char ch); //デリゲートの宣言(引数有、戻り値型無し)
class EventClass { //イベント発生を担うクラス
    public event Handler KeyHit; //イベントフィールド
    public void OnKeyHit(char ch) { //イベントを発生させるメソッド
        if (KeyHit != null) //イベントフィールドがヌルでなければ
            KeyHit(ch); //デリゲート経由で呼び出す
    }
}
class Show { //イベントで呼び出されるメソッドを持つクラス
    int sum = 0;
    public void keyshow(char ch) { //イベントで呼び出されるメソッド
        if (Char.IsDigit(ch)) { //10進数字か？
            int a = (int)char.GetNumericValue(ch); //実数に変換しintにキャスト
            sum += a; //合計に足し込む
            Console.WriteLine("+ {0}", a); //入力値を表示
            Console.WriteLine("= {0}", sum); //合計を表示
        } else if (ch == 'c') {
            sum = 0; //合計をゼロクリアする
            Console.WriteLine("合計がクリアされました");
        } else { //10進数字でもクリアでもなければ
            return; //何もしないで戻る
        }
    }
```

```
}
class event02 {
    public static void Main() {
        ConsoleKeyInfo cki; //キー情報構造体
        EventClass ec = new EventClass();  //イベント発生を担うクラスのインスタ
ンス生成
        Show s = new Show();//イベントで呼び出されるメソッドを持つクラスのインス
タンス生成
        ec.KeyHit += (c) => s.keyshow(c); //デリゲートにメソッドを登録。
        while (true) { //無限ループ
            if (Console.KeyAvailable) { //何かキーが押されている？
                cki = Console.ReadKey(true); //そのキーを得る
                if (cki.KeyChar == 'x') { //xキーならば
                    break; //抜ける＝プログラム終了
                } else {
                    ec.OnKeyHit(cki.KeyChar); //キー情報を渡してイベントを発生さ
せる
                }
            }
        }
    }
}
```

p.320 練習問題 ヒント

・練習問題１は、p.308 lambda02.csをそのまま用いると良い
・練習問題２はevent02.csを基にしてアレンジしよう
① デリゲートの定義はlambda02.csからコピー
② すると、イベントフィールドがnullの場合もreturnが必須になるので、-1を返すとしよう
③ Mainメソッドにおける整数の受け取りを２回にして、その和を表示したら終了とする
④ デリゲートに登録するメソッドを「(x, y) => { return x + y; }」にできるので、keyshowメソッドはShowクラスを含めて不要になる
⑤「10進数字か」のチェックと「実数に変換しintにキャスト」はMainメソッドに移すと良い

作成例

```
//p.320 練習問題2
using System;
delegate int Handler(int x, int y); //【変更】デリゲートの宣言(引数有、戻り値型
有)
class EventClass { //イベント発生を担うクラス
    public event Handler KeyHit; //イベントフィールド
    public int OnKeyHit(int x, int y) { //【変更】イベントを発生させるメソッド
        if (KeyHit != null) { //イベントフィールドがヌルでなければ
            return KeyHit(x, y); //【変更】デリゲート経由で呼び出す
        } else { //【以下追加】
            return -1;
        }
    }
}
class ex1202 {
    public static void Main() {
        ConsoleKeyInfo cki; //キー情報構造体
        EventClass ec = new EventClass();  //イベント発生を担うクラスのインスタ
ンス生成
        ec.KeyHit += (x, y) => { return x + y; }; //【変更】デリゲートに匿名メソ
```

```
ッドを登録。
        int temp = -1; //【追加】１値目
        while (true) { //無限ループ
            if (Console.KeyAvailable) { //何かキーが押されている？
                cki = Console.ReadKey(true); //そのキーを得る
                char ch = cki.KeyChar; //その文字を得る
                if (Char.IsDigit(ch)) { //【以下追加・変更】10進数字か？
                    int a = (int)char.GetNumericValue(ch); //実数に変換しintにキ
ャスト
                    if (temp == -1) { //１値目？
                        temp = a;
                    } else { //２値目？
                        int sum = ec.OnKeyHit(temp, a); //キー情報を渡してイベン
トを発生させる
                        Console.WriteLine("{0} + {1} = {2}", temp, a, sum);
                        break;
                    }
                }
            }
        }
    }
}
```

テキスト編次回予告：p.321「例外処理の基礎」から

ゲーム開発演習：自弾の発射と上移動、自弾の複数化、敵機の出現

演習26 画像の位置管理を中央座標に・改

・private void DrawImage(PaintEventArgs e, Image i, int x, int y)が冗長なので、
  private void DrawItem(PaintEventArgs e, Item it)としよう

作成例

```
//演習26 画像の位置管理を中央座標に・改
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
struct Item { //アイテムを表す構造体
    public Image i; //画像
    public int x;  //中心X座標
    public int y;  //中心Y座標
    public int hv; //左右方向の速度(左向きは負の数、右向きは正の数)
    public int v;  //表示状態(0：非表示、1以上：表示)
}
class Program : Form { //Formクラスの派生クラス
    [System.Runtime.InteropServices.DllImport("user32.dll")] //DLLインポート
    private static extern short GetKeyState(int nVirtKey); //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 playeri = Image.FromFile("player.gif"); //自機通常画像を読込む
    Image playerl = Image.FromFile("playerl.gif"); //自機左寄画像を読込む
    Image playerr = Image.FromFile("playerr.gif"); //自機右寄画像を読込む
    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
    Item player; //【変更】プレイヤーの構造体オブジェクト
    //【以下追加⇒変更】中央座標を用いる画像描画処理
    private void DrawItem(PaintEventArgs e, Item it) {
        int xx = it.x - it.i.Width / 2; //左上X座標を得る
        int yy = it.y - it.i.Height / 2; //左上Y座標を得る
        e.Graphics.DrawImage(it.i, xx, yy);
    }
    //描画処理のオーバライド
    protected override void OnPaint(PaintEventArgs e) {
        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
            switch (player.hv) { //【変更】自機の向きによって分岐
                case 0:  player.i = playeri; break; //【変更】通常画像にする
                case -1: player.i = playerl; break; //【変更】左寄画像にする
                case 1:  player.i = playerr; break; //【変更】右寄画像にする
            }
            DrawItem(e, player); //自機を描画
        }

    }
    //キー入力時処理
    void OnKeyDown(object o, KeyEventArgs e) {
        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
    }
    //タイマーイベント処理
    void Play(object o, EventArgs e) {
        backy = (backy + 1) % backi.Height; //1枚目の背景描画開始Y座標を下げる
        player.hv = 0; //【変更】自機の向きを無しにしておく
        if (player.x > playeri.Width / 2 && GetKeyState((int)Keys.Left) < 0) {
//【変更】範囲内で←キーが押されている？
            player.x -= 10; //【変更】自機を左へ
            player.hv = -1; //【変更】左向き
        }
        if (player.x < backi.Width - playeri.Width / 2 &&
GetKeyState((int)Keys.Right) < 0) { //【変更】範囲内で→キーが押されている？
```

```
            player.x += 10; //【変更】自機を右へ
            player.hv = 1; //【変更】右向き
        }
        if (GetKeyState((int)Keys.Up) < 0) { //↑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
    }
    //コンストラクタ
    Program() {
        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)
        player.i = playeri;  //【追加】自機の画像
        player.x = 320; //【移動・変更】自機の中心X座標
        player.y = 410; //【追加】自機の中心Y座標
        player.hv = 0;  //【移動・変更】自機の左右方向の速度
    }
    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
    }
}
```

演習27 自弾の出現（単独バージョン）

・ゲームモードが1(プレイ画面)の時に、スペースキーが押されたら自弾を出そう
・ただし、出現していないときに限る（まずは１個のみとする）
・出現位置は自機の位置で決まり、自機の直上とする
・自機はItem構造体で表す
・画像は下記を利用可能
  <img src=https://rundog.org/si/bullet.gif> bullet.gif 20x20

作成例

```
//演習27 自弾の出現（単独バージョン）
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
struct Item { //アイテムを表す構造体
    public Image i; //画像
    public int x;  //中心X座標
    public int y;  //中心Y座標
    public int hv; //左右方向の速度(左向きは負の数、右向きは正の数)
    public int v;  //表示状態(0：非表示、1以上：表示)
}
class Program : Form { //Formクラスの派生クラス
    [System.Runtime.InteropServices.DllImport("user32.dll")] //DLLインポート
    private static extern short GetKeyState(int nVirtKey); //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 playeri = Image.FromFile("player.gif"); //自機通常画像を読込む
```

```
    Image playerl = Image.FromFile("playerl.gif"); //自機左寄画像を読込む
    Image playerr = Image.FromFile("playerr.gif"); //自機右寄画像を読込む
    Image bulleti = Image.FromFile("bullet.gif"); //【追加】自弾画像を読込む
    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
    Item player; //自機の構造体オブジェクト
    Item pb; //【追加】自弾の構造体オブジェクト
    //中央座標を用いる画像描画処理
    private void DrawItem(PaintEventArgs e, Item it) {
        int xx = it.x - it.i.Width / 2; //左上X座標を得る
        int yy = it.y - it.i.Height / 2; //左上Y座標を得る
        e.Graphics.DrawImage(it.i, xx, yy);
    }
    //描画処理のオーバライド
    protected override void OnPaint(PaintEventArgs e) {
        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
            switch (player.hv) { //自機の向きによって分岐
                case 0:  player.i = playeri; break; //通常画像にする
                case -1: player.i = playerl; break; //左寄画像にする
                case 1:  player.i = playerr; break; //右寄画像にする
            }
            DrawItem(e, player); //自機を描画
            if (pb.v == 1) { //【以下追加】自弾がある？
                DrawItem(e, pb); //自弾を描画
            }
        }

    }
    //キー入力時処理
    void OnKeyDown(object o, KeyEventArgs e) {
        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
    }
```

```
    //タイマーイベント処理
    void Play(object o, EventArgs e) {
        backy = (backy + 1) % backi.Height; //1枚目の背景描画開始Y座標を下げる
        player.hv = 0; //自機の向きを無しにしておく
        if (player.x > playeri.Width / 2 && GetKeyState((int)Keys.Left) < 0) {
//範囲内で←キーが押されている？
            player.x -= 10; //自機を左へ
            player.hv = -1; //左向き
        }
        if (player.x < backi.Width - playeri.Width / 2 &&
GetKeyState((int)Keys.Right) < 0) { //範囲内で→キーが押されている？
            player.x += 10; //自機を右へ
            player.hv = 1; //右向き
        }
        if (GetKeyState((int)Keys.Space) < 0) { //【以下追加】スペースキーが押さ
れている？
            if (pb.v == 0) { //自弾が非表示？
                pb.v = 1; //表示にする
                pb.i = bulleti; //画像
                pb.x = player.x; //X座標は自機と同じ
                pb.y = player.y - player.i.Height / 2 - pb.i.Height / 2; //Y座標
は上
            }
        }
        Invalidate(); //画面再描画を依頼
    }
    //コンストラクタ
    Program() {
        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)
        player.i = playeri;  //自機の画像
        player.x = 320; //自機の中心X座標
        player.y = 410; //自機の中心Y座標
        player.hv = 0;  //自機の左右方向の速度
    }
    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
    }
}
```

演習28 自弾の上移動(単独バージョン)

・自弾が出現状態であれば、上へ移動しよう
・完全に見えなくなったら、出現状態を無に戻して再発射可能にしよう
・上下方向の移動速度vvをItem構造体に追加しよう(値は適当に)

作成例

```
//演習28 自弾の上移動(単独バージョン)
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
struct Item { //アイテムを表す構造体
    public Image i; //画像
    public int x;  //中心X座標
    public int y;  //中心Y座標
    public int hv; //左右方向の速度(左向きは負の数、右向きは正の数)
    public int vv; //【追加】上下方向の速度(上向きは負の数、下向きは正の数)
    public int v;  //表示状態(0：非表示、1以上：表示)
}
class Program : Form { //Formクラスの派生クラス
    [System.Runtime.InteropServices.DllImport("user32.dll")] //DLLインポート
    private static extern short GetKeyState(int nVirtKey); //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 playeri = Image.FromFile("player.gif"); //自機通常画像を読込む
    Image playerl = Image.FromFile("playerl.gif"); //自機左寄画像を読込む
    Image playerr = Image.FromFile("playerr.gif"); //自機右寄画像を読込む
    Image bulleti = Image.FromFile("bullet.gif"); //自弾画像を読込む
    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
    Item player; //自機の構造体オブジェクト
    Item pb; //自弾の構造体オブジェクト
    //中央座標を用いる画像描画処理
    private void DrawItem(PaintEventArgs e, Item it) {
        int xx = it.x - it.i.Width / 2; //左上X座標を得る
        int yy = it.y - it.i.Height / 2; //左上Y座標を得る
        e.Graphics.DrawImage(it.i, xx, yy);
    }
    //描画処理のオーバライド
    protected override void OnPaint(PaintEventArgs e) {
        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
            switch (player.hv) { //自機の向きによって分岐
                case 0:  player.i = playeri; break; //通常画像にする
                case -1: player.i = playerl; break; //左寄画像にする
                case 1:  player.i = playerr; break; //右寄画像にする
            }
            DrawItem(e, player); //自機を描画
            if (pb.v == 1) { //自弾がある？
                DrawItem(e, pb); //自弾を描画
```

```
            }
        }

    }
    //キー入力時処理
    void OnKeyDown(object o, KeyEventArgs e) {
        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
    }
    //タイマーイベント処理
    void Play(object o, EventArgs e) {
        backy = (backy + 1) % backi.Height; //1枚目の背景描画開始Y座標を下げる
        player.hv = 0; //自機の向きを無しにしておく
        if (player.x > playeri.Width / 2 && GetKeyState((int)Keys.Left) < 0) {
//範囲内で←キーが押されている？
            player.x -= 10; //自機を左へ
            player.hv = -1; //左向き
        }
        if (player.x < backi.Width - playeri.Width / 2 &&
GetKeyState((int)Keys.Right) < 0) { //範囲内で→キーが押されている？
            player.x += 10; //自機を右へ
            player.hv = 1; //右向き
        }
        if (GetKeyState((int)Keys.Space) < 0) { //スペースキーが押されている？
            if (pb.v == 0) { //自弾が非表示？
                pb.v = 1; //表示にする
                pb.i = bulleti; //画像
                pb.x = player.x; //X座標は自機と同じ
                pb.y = player.y - player.i.Height / 2 - pb.i.Height / 2; //Y座標
は自機の直上
                pb.vv = -5; //【追加】上移動速度
            }
        }
        if (pb.v != 0) { //【以下追加】自弾が存在？
            pb.y += pb.vv; //上へ移動
            if (pb.y + pb.i.Height / 2 < 0) { //画面上端より上に出たら
                pb.v = 0; //自弾を消す
            }
        }
        Invalidate(); //画面再描画を依頼
    }
    //コンストラクタ
    Program() {
        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)
        player.i = playeri;  //自機の画像
        player.x = 320; //自機の中心X座標
        player.y = 410; //自機の中心Y座標
        player.hv = 0;  //自機の左右方向の速度
```

```
    }
    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
    }
}
```

演習29 自弾のアニメーション

・自弾を左右反転した画像と交互に表示することで、回転しているように見せよう
・画像は下記を利用可能
  <img src=https://rundog.org/si/bullet2.gif> bullet2.gif 20x20

提出：演習29(未完成、演習28でも可)

ゲーム開発演習次回予告：自弾の複数化、敵機の出現