Скопировал в ПДФ-файл текст одного из ключевых классов игры.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using AstroShooterGame.Sprites;
using Microsoft.Xna.Framework.Input;
using GameGame.Managers;
using GameGame.Models;

namespace GameGame.States
{
  public class GameState : State
  {
    private EnemyManager _enemyManager;

    private SpriteFont _font;

    private List<Player> _players;

    private ScoreManager _scoreManager;

    private List<Sprite> _sprites;

    public int PlayerCount;

    public GameState(Game1 game, ContentManager content)
      : base(game, content)
    {
    }

    public override void LoadContent()
    {
            var playerTexture = _content.Load<Texture2D>("Player");

            var bulletTexture = _content.Load<Texture2D>("Bullet");

      _font = _content.Load<SpriteFont>("Font");

      _scoreManager = ScoreManager.Load();

      _sprites = new List<Sprite>()
      {
        new Sprite(_content.Load<Texture2D>("Game"))
        {
          Layer = 0.0f,
          Position = new Vector2(Game1.ScreenWidth / 2, Game1.ScreenHeight / 2),
        }
      };

      var bulletPrefab = new Bullet(bulletTexture)
      {
        Explosion = new Explosion(new Dictionary<string, Models.Animation>()
            {
              {
                "Explode", new Models.Animation(_content.Load<Texture2D>("Explosion"),
6)
                { FrameSpeed = 0.1f, }
              }
            })
```

```
        {
          Layer = 0.5f,
        }
      };

      if (PlayerCount >= 1)
      {
        _sprites.Add(new Player(playerTexture)
        {
          Colour = Color.Green,
          Position = new Vector2(100, Game1.ScreenHeight / 2),
          Layer = 0.3f,
          Bullet = bulletPrefab,
          Input = new Models.Input()
          {
            Up = Keys.Up,
            Down = Keys.Down,
            Left = Keys.Left,
            Right = Keys.Right,
            Shoot = Keys.Space,
          },
          Health = 10,
          Score = new Models.Score()
          {
            PlayerName = "Player 1",
            Value = 0,
          },
        });
      }

      _players = _sprites.Where(c => c is Player).Select(c => (Player)c).ToList();

      _enemyManager = new EnemyManager(_content)
      {
        Bullet = bulletPrefab,
        Explosion = new Explosion(new Dictionary<string, Models.Animation>()
        {
           {
                "Explode", new Models.Animation(_content.Load<Texture2D>("Explosion"),
6)
                { FrameSpeed = 0.1f, }
           }
        })
        {
            Layer = 0.5f,
        }

      };
    }

    public override void Update(GameTime gameTime)
    {
      if (Keyboard.GetState().IsKeyDown(Keys.Escape))
        _game.ChangeState(new MenuState(_game, _content));

      foreach (var sprite in _sprites)
        sprite.Update(gameTime);

      _enemyManager.Update(gameTime);
      if (_enemyManager.CanAdd && _sprites.Where(c => c is Enemy).Count() <
_enemyManager.MaxEnemies)
      {
        _sprites.Add(_enemyManager.GetEnemy());
      }
    }
```

```
public override void PostUpdate(GameTime gameTime)
{
  var collidableSprites = _sprites.Where(c => c is ICollidable);

  foreach (var spriteA in collidableSprites)
  {
    foreach (var spriteB in collidableSprites)
    {
      if (spriteA == spriteB)
        continue;

      if (!spriteA.CollisionArea.Intersects(spriteB.CollisionArea))
        continue;

      if (spriteA.Intersects(spriteB))
        ((ICollidable)spriteA).OnCollide(spriteB);
    }
  }


  int spriteCount = _sprites.Count;
  for (int i = 0; i < spriteCount; i++)
  {
    var sprite = _sprites[i];
    foreach (var child in sprite.Children)
      _sprites.Add(child);

    sprite.Children = new List<Sprite>();
  }

  for (int i = 0; i < _sprites.Count; i++)
  {
    if (_sprites[i].IsRemoved)
    {
      _sprites.RemoveAt(i);
      i--;
    }
  }


  if (_players.All(c => c.IsDead))
  {
    foreach (var player in _players)
      _scoreManager.Add(player.Score);

    ScoreManager.Save(_scoreManager);

    _game.ChangeState(new HighscoresState(_game, _content));
  }
}

public override void Draw(GameTime gameTime, SpriteBatch spriteBatch)
{
  spriteBatch.Begin(SpriteSortMode.FrontToBack);

  foreach (var sprite in _sprites)
    sprite.Draw(gameTime, spriteBatch);

  spriteBatch.End();

  spriteBatch.Begin();

  float x = 10f;
  foreach (var player in _players)
```

```
      {
        spriteBatch.DrawString(_font, "Player: " + player.Score.PlayerName, new
Vector2(x, 10f), Color.White);
        spriteBatch.DrawString(_font, "Health: " + player.Health, new Vector2(x, 30f),
Color.White);
        spriteBatch.DrawString(_font, "Score: " + player.Score.Value, new Vector2(x,
50f), Color.White);

        x += 150;
      }
      spriteBatch.End();
    }
  }
}
```