Ročníková práce

Mario

Ročníková práce

Vedoucí práce: Vypracoval: David Knieradl

Mgr. Skalka František Studijní obor: IT

Číslo oboru: 18-20-M/01

Třída: IT3

Prohlašuji, že jsem ročníkovou práci vypracoval samostatně s přispěním vedoucího práce a použil jsem jen literaturu a informační zdroje uvedené v kapitole literatura.

Děkuji Mgr. Františku Skalkovi za odborné vedení a cenné rady, které mi poskytl při zpracování této ročníkové práce.

Souhlasím s půjčováním a zpřístupněním ročníkové práce.

V Brně 14. května 2017

Obsah

[1 Úvod 5](#_Toc387158007)

[2 Stať – vlastní kapitoly práce 5](#_Toc387158008)

[2.1 Nadpis druhé úrovně 5](#_Toc387158009)

[3 Závěr 5](#_Toc387158010)

[4 Literatura 5](#_Toc387158011)

[Přílohy 5](#_Toc387158012)

[A Název přílohy 5](#_Toc387158013)

# Úvod

Tato ročníková práce se zabývá tématem „Tabulky“. Jelikož je téma „Tabulky“, tak tento program samozřejmě využívá různé druhy polí. Nejčastěji se v něm setkáte s poli jednorozměrnými. Jedná se pouze o program ve velice rané fázi vývoje. Napsán byl formou OOP, neboli objektově orientovaným programováním. Ve stručnosti se jedná o takový druh programování, kdy vytváříme několik různých objektů, se kterými následně pracujeme. Podle názvu je poznat, že jsem přetvořil jednu z celosvětově známých her a to hru firmy Nintendo „Mario“. Obsahuje však pouze jeden svět, hlavní menu a stránku o programu. Největší část programu však tvoří jádro, které jsem během roku 2016/2017 stvořil. Lze jej nalézt na mém GitHubovém profilu (meowside) pod názvem DKEngine ([www.github.com/meowside/DKEngine](http://www.github.com/meowside/DKEngine)). Zde naleznete hodiny velice záživného čtení. Celá ročníková práce včetně jádra DKEngine čítá několik tisíc řádku. Při spuštění programu je potřeba jádro nejdříve inicializovat, až poté lze s ním pracovat podle potřeby. Pohyb postavy probíhá pomocí kláves „WASD“ a „Space“. Klávesa „W“ slouží pro skákání postavy. V momentě, kdy je nad postavou jiný objekt s komponentou „Collider“, tak se o tento objekt postava zastaví. Takto funguje kolize do strany i levé, pravé a spodní. V momentě střetu s nepřítelem se rozhoduje, do které strany byl hráč zasažen. Pokud se jednalo o stranu spodní, nepřítel byl poražen a přičte se skóre. V opačném případě přechází hráč do stavu o jedno nižší. Struktura takovýchto stavů je následovná.

Invincible -> Fire -> Super -> Small -> Dead

V tento moment je maximální funkčí stav „Super“. Do tohoto stavu se lze dostat pomocí objektu „PowerUp“, který vytvoří na mapě pohybující se houbu. Po získání této houby se přehraje animace zvětšení postavy a příčtou se body za získání houby.

Systém skóre je v tuto chvíli velice jednoduchý. Za přemožení nepřítele „Goomba“ lze získat bodů 100, za získání houby 200, za získání penízku 100. Momentálně ve hře nfunguje kombo systém, který by násobil skóre pomocí v sérii zabitých nepřátel.

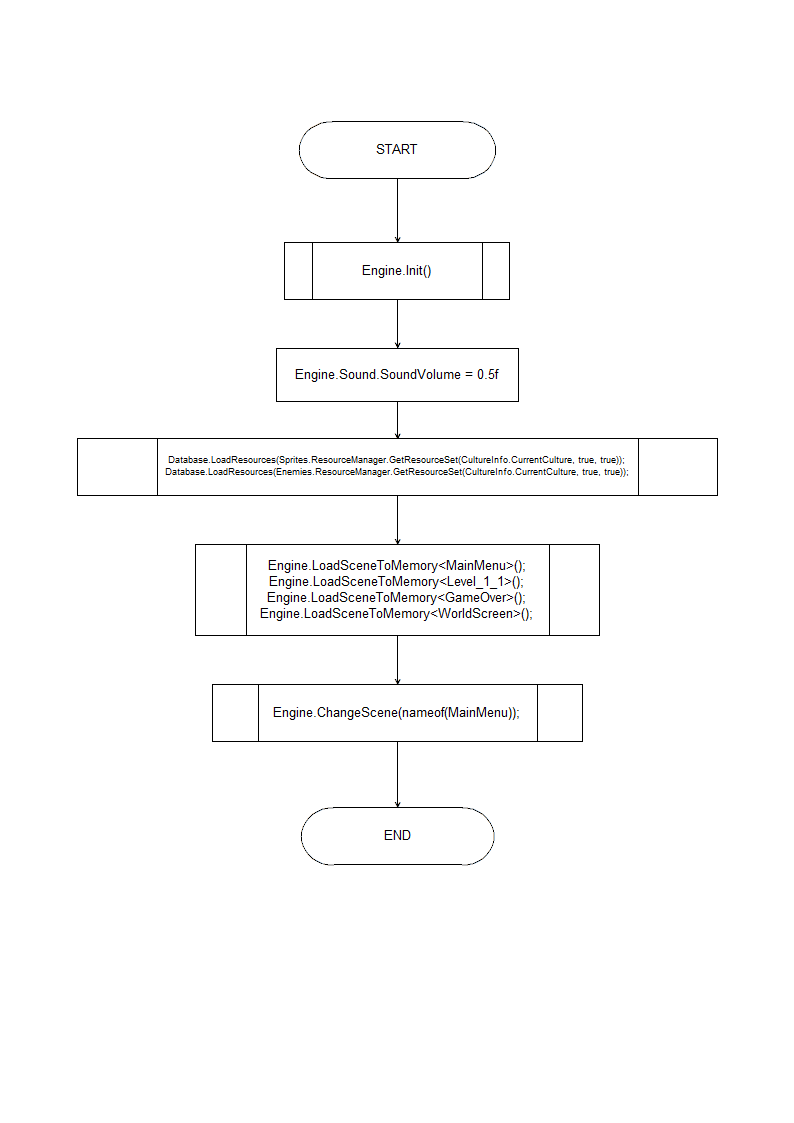
Program dále využívá knihovny NAudio. Jedná se o knihovnu zaměřenou na přehrávání audio záznamů. Pomocí této knihovny jsem vytvočil zvukový systém shopný přehrávat několik zvukových efektů zároveň v reálném čase.

Animace jsou následně prováděny pomocí visuálních prvků, kterými jsou obrázky ve formátu „PNG“ nebo „GIF“. K animování slouží objekt „Animator“ dostupný v jádře DKEngine.

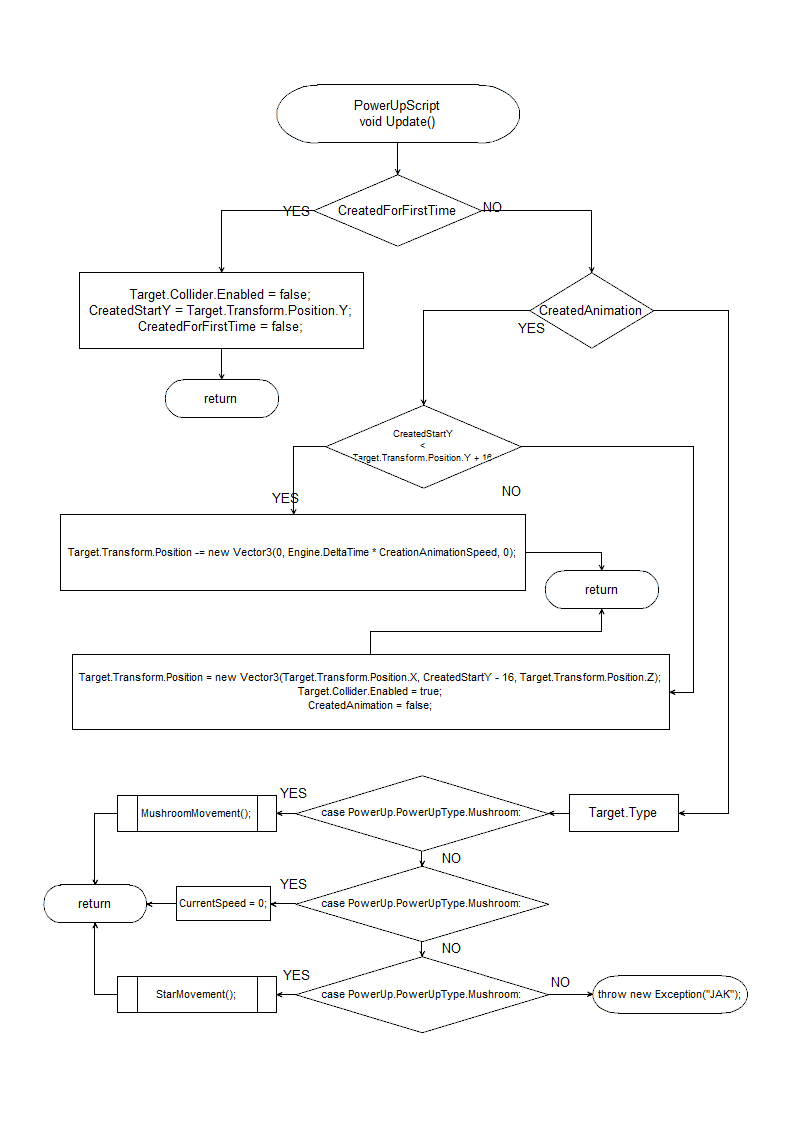
# Stať

## Vývojový diagram

### Metoda „Main“



### PowerUpScript – metoda Update



## Zdrojový kód

### DKEngine

#### Engine.cs

1. /\*\*
2. \* (C) 2017 David Knieradl
3. \*
4. \* For the brave souls who get this far: You are the chosen ones,
5. \* the valiant knights of programming who toil away, without rest,
6. \* fixing our most awful code. To you, true saviors, kings of men,
7. \* I say this: never gonna give you up, never gonna let you down,
8. \* never gonna run around and desert you. Never gonna make you cry,
9. \* never gonna say goodbye. Never gonna tell a lie and hurt you.
10. \*/
11. using DKEngine.Core;
12. using DKEngine.Core.Components;
13. using DKEngine.Core.Ext;
14. using DKEngine.Core.UI;
15. using DKEngine.Data;
16. using System;
17. using System.Collections.Generic;
18. using System.Diagnostics;
19. using System.Drawing;
20. using System.Linq;
21. using System.Runtime.InteropServices;
22. using System.Threading;
23. using System.Threading.Tasks;
24. namespace DKEngine
25. {
26. /// <summary>
27. /// Engine class
28. /// </summary>
29. public static class Engine
30. {
31. /// <summary>
32. /// Sound subclass
33. /// </summary>
34. public static class Sound
35. {
36. /// <summary>
37. /// Enables sound
38. /// </summary>
39. public static bool IsSoundEnabled = true;
40. /// <summary>
41. /// Sets volume on sound inicialization
42. /// </summary>
43. public static float SoundVolume = 1f;
44. internal readonly static SoundPlayer Instance = new SoundPlayer();
45. }
46. /// <summary>
47. /// Render subclass
48. /// </summary>
49. public static class Render
50. {
51. /// <summary>
52. /// Sets resolution scale in %
53. /// </summary>
54. public const int ResolutionScale = 50;
55. /// <summary>
56. /// The resolution ratio
57. /// </summary>
58. public const float ResolutionRatio = ResolutionScale / 100f;
59. /// <summary>
60. /// The rendered image width
61. /// </summary>
62. public const int RenderWidth = (int)(640 \* ResolutionRatio);
63. /// <summary>
64. /// The rendered image height
65. /// </summary>
66. public const int RenderHeight = (int)(480 \* ResolutionRatio);
67. internal const int ImageBufferSize = 3 \* RenderWidth \* RenderHeight;
68. internal const int ImageKeyBufferSize = RenderWidth \* RenderHeight;
69. internal static byte[] imageBuffer;
70. internal static byte[] imageBufferKey;
71. internal static byte[] ImageOutData;
72. internal static bool AbortRender = false;
73. internal const int Limiter = 1000;
74. }
75. /// <summary>
76. /// Input subclass
77. /// </summary>
78. public static class Input
79. {
80. [DllImport("user32.dll")]
81. private static extern ushort GetKeyState(short nVirtKey);
82. private const ushort keyDownBit = 0x80;
83. internal static bool[] KeysPressed;
84. internal static bool[] KeysDown;
85. internal static bool[] KeysReleased;
86. internal static bool[] KeysUp;
87. internal static short NumberOfKeys;
88. /// <summary>
89. /// Determines whether [is key pressed] [the specified key].
90. /// </summary>
91. /// <param name="key">The key</param>
92. /// <returns>
93. /// <c>true</c> if [is key pressed] [the specified key]; otherwise, <c>false</c>.
94. /// </returns>
95. public static bool IsKeyPressed(ConsoleKey key)
96. {
97. return KeysPressed[(short)key];
98. }
99. /// <summary>
100. /// Determines whether [is key down] [the specified key].
101. /// </summary>
102. /// <param name="key">The key</param>
103. /// <returns>
104. /// <c>true</c> if [is key down] [the specified key]; otherwise, <c>false</c>.
105. /// </returns>
106. public static bool IsKeyDown(ConsoleKey key)
107. {
108. return KeysDown[(short)key];
109. }
110. /// <summary>
111. /// Determines whether [is key up] [the specified key].
112. /// </summary>
113. /// <param name="key">The key</param>
114. /// <returns>
115. /// <c>true</c> if [is key up] [the specified key]; otherwise, <c>false</c>.
116. /// </returns>
117. public static bool IsKeyUp(ConsoleKey key)
118. {
119. return KeysUp[(short)key];
120. }
121. /// <summary>
122. /// Determines whether [is key released] [the specified key].
123. /// </summary>
124. /// <param name="key">The key</param>
125. /// <returns>
126. /// <c>true</c> if [is key released] [the specified key]; otherwise, <c>false</c>.
127. /// </returns>
128. public static bool IsKeyReleased(ConsoleKey key)
129. {
130. return KeysReleased[(short)key];
131. }
132. internal static void CheckForKeys()
133. {
134. for (int key = 0; key < NumberOfKeys; key++)
135. {
136. bool IsDown = ((GetKeyState((short)key) & keyDownBit) == keyDownBit);
137. if (IsDown)
138. {
139. if (!KeysDown[key])
140. {
141. KeysUp[key] = false;
142. KeysReleased[key] = false;
143. KeysPressed[key] = true;
144. KeysDown[key] = true;
145. }
146. else if (KeysPressed[key])
147. {
148. KeysPressed[key] = false;
149. }
150. }
151. else
152. {
153. if (KeysDown[key])
154. {
155. KeysPressed[key] = false;
156. KeysDown[key] = false;
157. KeysReleased[key] = true;
158. KeysUp[key] = true;
159. }
160. else if (KeysReleased[key])
161. {
162. KeysReleased[key] = false;
163. }
164. }
165. }
166. }
167. }
168. private static bool \_IsInitialised = false;
169. private static Thread BackgroundWorks;
170. private static TextBlock FpsMeter;
171. private static Stopwatch DeltaT;
172. internal static Camera BaseCam;
173. internal static Scene CurrentScene { get; set; }
174. internal static Scene LoadingScene { get; set; }
175. internal static Type LoadingSceneType { get; set; }
176. internal static List<GameObject> RenderObjects;
177. private static float deltaT = 0;
178. public static float DeltaTime { get { return deltaT; } }
179. private static TimeSpan lastUpdated = new TimeSpan();
180. public static TimeSpan LastUpdated { get { return lastUpdated; } }
181. public static string SceneName { get { return Engine.LoadingScene != null ? Engine.LoadingScene.Name : ""; } }
182. private static readonly TimeSpan \_firstTimeLoadDelay = new TimeSpan(0, 0, 1);
183. private static TimeSpan FirstTimeLoadDelay = new TimeSpan();
184. private static bool FirstTimeLoaded = true;
185. internal static event EngineHandler UpdateEvent;
186. internal delegate void EngineHandler();
187. /// <summary>
188. /// Sets engine to work.
189. /// </summary>
190. /// <exception cref="System.Exception">
191. /// Engine initialisation failed\n" + e
192. /// or
193. /// Engine is being initialised second time
194. /// </exception>
195. public static void Init()
196. {
197. if (!\_IsInitialised)
198. {
199. try
200. {
201. WindowControl.WindowInit();
202. Database.InitDatabase();
203. Render.imageBuffer = new byte[Render.ImageBufferSize];
204. Render.imageBufferKey = new byte[Render.ImageKeyBufferSize];
205. Render.ImageOutData = new byte[Render.ImageBufferSize];
206. Input.NumberOfKeys = (short)Enum.GetNames(typeof(ConsoleKey)).Length;
207. Input.KeysPressed = new bool[Input.NumberOfKeys];
208. Input.KeysDown = new bool[Input.NumberOfKeys];
209. Input.KeysUp = new bool[Input.NumberOfKeys];
210. Input.KeysReleased = new bool[Input.NumberOfKeys];
211. DeltaT = Stopwatch.StartNew();
212. RenderObjects = new List<GameObject>(0xFFFF);
213. //Sound.OutputDevice = new WaveOut();
214. FpsMeter = new TextBlock();
215. FpsMeter.Transform.Position = new Vector3(4, -4, 128);
216. FpsMeter.Transform.Dimensions = new Vector3(50, 5, 1);
217. FpsMeter.VAlignment = Text.VerticalAlignment.Bottom;
218. FpsMeter.HAlignment = Text.HorizontalAlignment.Left;
219. FpsMeter.Text = "0";
220. FpsMeter.IsGUI = true;
221. FpsMeter.TextShadow = true;
222. FpsMeter.Foreground = Color.FromArgb(0xFF, 0x00, 0xFF, 0xFF);
223. FpsMeter.InitInternal();
224. UpdateEvent += FpsMeter.Scripts[0].UpdateHandle;
225. BackgroundWorks = new Thread(Update);
226. //RenderWorker = new Thread(RenderImage);
227. BackgroundWorks.Start();
228. //RenderWorker.Start();
229. #if !DEBUG
230. SplashScreen();
231. #endif
232. \_IsInitialised = true;
233. }
234. catch (Exception e)
235. {
236. throw new Exception("Engine initialisation failed\n" + e);
237. }
238. }
239. else
240. throw new Exception("Engine is being initialised second time");
241. }
242. public static void LoadSceneToMemory<T>(object[] argsPreLoad = null, object[] argsPostLoad = null)
243. where T : Scene
244. {
245. Engine.LoadingScene = (T)Activator.CreateInstance(typeof(T));
246. Engine.LoadingScene.argsPreLoad = argsPreLoad;
247. Engine.LoadingScene.argsPostLoad = argsPostLoad;
248. Engine.LoadingScene.Set(argsPreLoad);
249. Engine.LoadingScene.Init();
250. Database.AddScene(Engine.LoadingScene);
251. }
252. /// <summary>
253. /// Loads the scene to memory.
254. /// </summary>
255. /// <typeparam name="T">Scene</typeparam>
256. /// <param name="argsPreLoad">The arguments pre load.</param>
257. /// <param name="argsPostLoad">The arguments post load.</param>
258. public static void LoadScene<T>(object[] argsPreLoad = null, object[] argsPostLoad = null) where T : Scene
259. {
260. LoadingSceneType = typeof(T);
261. LoadScene(LoadingSceneType, argsPreLoad, argsPostLoad);
262. }
263. public static void LoadScene(Type scene, object[] argsPreLoad = null, object[] argsPostLoad = null)
264. {
265. if (!scene.IsSubclassOf(typeof(Scene)))
266. throw new Exception($"Provided type {scene} is not subclass of Scene");
267. Engine.LoadingScene = (Scene)Activator.CreateInstance(LoadingSceneType);
268. Engine.LoadingScene.argsPreLoad = argsPreLoad;
269. Engine.LoadingScene.argsPostLoad = argsPostLoad;
270. Engine.LoadingScene.Set(argsPreLoad);
271. Engine.LoadingScene.Init();
272. UnregisterScene();
273. RegisterScene(Engine.LoadingScene, argsPostLoad);
274. }
275. /// <summary>
276. /// Reloads the scene.
277. /// </summary>
278. /// <param name="Name">The name of scene</param>
279. public static void ReloadScene(string Name, object[] argsPreLoad = null)
280. {
281. Database.RewriteWorld(Name, argsPreLoad);
282. }
283. /// <summary>
284. /// Changes the scene.
285. /// </summary>
286. /// <param name="Name">The name</param>
287. /// <param name="Reload">if set to <c>true</c> [reload]</param>
288. /// <param name="args">The arguments</param>
289. public static void ChangeScene(string Name, bool Reload = false, object[] argsPreLoad = null, object[] argsPostLoad = null)
290. {
291. UnregisterScene();
292. if (Reload)
293. {
294. ReloadScene(Name, argsPreLoad);
295. if (argsPostLoad != null)
296. Database.GetScene(Name).argsPostLoad = argsPostLoad;
297. }
298. RegisterScene(Database.GetScene(Name), argsPostLoad);
299. }
300. private static void UnregisterScene()
301. {
302. try
303. {
304. Engine.CurrentScene.Unload();
305. foreach (var item in CurrentScene.AllBehaviors)
306. {
307. try
308. {
309. UpdateEvent -= item.UpdateHandle;
310. }
311. catch { }
312. }
313. while (CurrentScene.GameObjectsAddedToRender.Count > 0)
314. {
315. GameObject tmp = CurrentScene.GameObjectsAddedToRender.Pop();
316. if (Engine.RenderObjects.Contains(tmp))
317. Engine.RenderObjects.Remove(tmp);
318. CurrentScene.GameObjectsToAddToRender.Push(tmp);
319. }
320. }
321. catch { }
322. }
323. private static void RegisterScene(Scene source, object[] args)
324. {
325. Engine.LoadingScene = source;
326. if (args != null)
327. {
328. source.argsPostLoad = args;
329. source.Set(args);
330. }
331. else if (source.argsPostLoad != null)
332. source.Set(source.argsPostLoad);
333. foreach (var item in source.AllBehaviors)
334. {
335. try
336. {
337. UpdateEvent += item.UpdateHandle;
338. }
339. catch { }
340. }
341. Engine.BaseCam = Engine.LoadingScene.BaseCamera;
342. Engine.CurrentScene = source;
343. }
344. public static void ReloadCurrentScene()
345. {
346. UnregisterScene();
347. LoadScene(LoadingSceneType);
348. }
349. private static void SplashScreen()
350. {
351. if (!\_IsInitialised)
352. {
353. Engine.LoadScene<SplashScreenScene>();
354. SpinWait.SpinUntil(() => ((SplashScreenScene)Engine.CurrentScene).Splash.Animator.NumberOfPlays >= 1);
355. }
356. }
357. private static void Update()
358. {
359. Task imageRender = Task.Factory.StartNew(RenderImage);
360. int NumberOfFrames = 0;
361. TimeSpan timeOut = new TimeSpan(0, 0, 0, 0, 500);
362. Stopwatch time = Stopwatch.StartNew();
363. Stopwatch fpsLimiter = Stopwatch.StartNew();
364. while (true)
365. {
366. Input.CheckForKeys();
367. lastUpdated += DeltaT.Elapsed;
368. deltaT = (float)DeltaT.Elapsed.TotalSeconds;
369. DeltaT?.Restart();
370. if (!FirstTimeLoaded)
371. {
372. UpdateEvent?.Invoke();
373. while (Engine.CurrentScene?.NewlyGeneratedComponents.Count > 0)
374. {
375. Engine.CurrentScene.NewlyGeneratedComponents.Pop().InitInternal();
376. }
377. while (Engine.CurrentScene?.NewlyGeneratedBehaviors.Count > 0)
378. {
379. Behavior tmp = Engine.CurrentScene.NewlyGeneratedBehaviors.Pop();
380. UpdateEvent += tmp.UpdateHandle;
381. tmp.Start();
382. }
383. while (Engine.CurrentScene?.DestroyObjectAwaitList.Count > 0)
384. {
385. GameObject tmp = Engine.CurrentScene.DestroyObjectAwaitList[0];
386. Engine.CurrentScene.DestroyObjectAwaitList.RemoveAt(0);
387. tmp.Destroy();
388. }
389. while (Engine.CurrentScene?.GameObjectsToAddToRender.Count > 0)
390. {
391. GameObject tmp = Engine.CurrentScene.GameObjectsToAddToRender.Pop();
392. Engine.RenderObjects.Add(tmp);
393. Engine.CurrentScene.GameObjectsAddedToRender.Push(tmp);
394. }
395. List<GameObject> reference = Engine.RenderObjects.GetGameObjectsInView();
396. if (Engine.CurrentScene != null)
397. {
398. List<Collider> VisibleTriggers = Engine.CurrentScene?.AllGameObjectsColliders.Where(obj => obj.IsTrigger).ToList();
399. List<Collider> VisibleColliders = Engine.CurrentScene?.AllGameObjectsColliders.Where(obj => !obj.IsTrigger).ToList();
400. int ColliderCount = VisibleTriggers.Count;
401. for (int i = 0; i < ColliderCount; i++)
402. VisibleTriggers[i]?.TriggerCheck(VisibleColliders);
403. }
404. Engine.CurrentScene?.BaseCamera?.BufferImage(reference);
405. Buffer.BlockCopy(Render.imageBuffer, 0, Render.ImageOutData, 0, Render.ImageBufferSize);
406. }
407. else
408. {
409. FirstTimeLoadDelay += new TimeSpan(0, 0, 0, 0, (int)(DeltaTime \* 1000));
410. if (FirstTimeLoadDelay > \_firstTimeLoadDelay)
411. {
412. FirstTimeLoaded = false;
413. FirstTimeLoadDelay = new TimeSpan();
414. }
415. }
416. NumberOfFrames++;
417. Vsync(Render.Limiter, (int)fpsLimiter.ElapsedMilliseconds);
418. fpsLimiter.Restart();
419. if (time.ElapsedMilliseconds > timeOut.TotalMilliseconds)
420. {
421. long t = NumberOfFrames \* 1000 / time.ElapsedMilliseconds;
422. FpsMeter.Text = t.ToString();
423. /\*#if DEBUG
424. Debug.WriteLine(t);
425. #endif\*/
426. time.Restart();
427. NumberOfFrames = 0;
428. }
429. }
430. }
431. private static async void RenderImage()
432. {
433. IntPtr ConsoleWindow = GetConsoleWindow();
434. using (Graphics g = Graphics.FromHwnd(ConsoleWindow))
435. {
436. g.CompositingQuality = System.Drawing.Drawing2D.CompositingQuality.HighSpeed;
437. g.PixelOffsetMode = System.Drawing.Drawing2D.PixelOffsetMode.HighSpeed;
438. g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.None;
439. g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.NearestNeighbor;
440. Rectangle Screen = System.Windows.Forms.Screen.FromHandle(ConsoleWindow).Bounds;
441. int Width = Screen.Width;
442. int Height = Screen.Height;
443. float ScaleRatio = Height / Engine.Render.RenderHeight;
444. int RasteredHeight = (int)(Engine.Render.RenderHeight \* ScaleRatio);
445. int RasteredWidth = (int)(Engine.Render.RenderWidth \* ScaleRatio);
446. int XOffset = (int)(Width - RasteredWidth) / 2;
447. int YOffset = (int)(Height - RasteredHeight) / 2;
448. while (!Render.AbortRender)
449. {
450. Rectangle ScreenResCheck = System.Windows.Forms.Screen.FromHandle(ConsoleWindow).Bounds;
451. if (ScreenResCheck != Screen)
452. {
453. Width = ScreenResCheck.Width;
454. Height = ScreenResCheck.Height;
455. XOffset = (int)(Width - (Engine.Render.RenderWidth \* ScaleRatio)) / 2;
456. YOffset = (int)(Height - (Engine.Render.RenderHeight \* ScaleRatio)) / 2;
457. }
458. unsafe
459. {
460. fixed (byte\* ptr = Render.ImageOutData)
461. {
462. using (Bitmap outFrame = new Bitmap(Render.RenderWidth,
463. Render.RenderHeight,
464. 3 \* Render.RenderWidth,
465. System.Drawing.Imaging.PixelFormat.Format24bppRgb,
466. new IntPtr(ptr)))
467. {
468. Rectangle imageRect = new Rectangle(XOffset,
469. YOffset,
470. RasteredWidth,
471. RasteredHeight);
472. g.DrawImage(outFrame, imageRect);
473. }
474. }
475. }
476. await Task.Delay(1);
477. }
478. }
479. }
480. private static void Vsync(int TargetFrameRate, int ImageRenderDelay)
481. {
482. int targetDelay = 1000 / TargetFrameRate;
483. if (ImageRenderDelay < targetDelay)
484. {
485. Thread.Sleep(targetDelay - ImageRenderDelay);
486. }
487. }
488. [DllImport("kernel32.dll", SetLastError = true)]
489. private static extern IntPtr GetConsoleWindow();
490. }
491. }

#### Core/Components/AnimationNode.cs

1. namespace DKEngine.Core.Components
2. {
3. /// <summary>
4. /// Node used in Animator Component
5. /// </summary>
6. /// <seealso cref="DKEngine.Core.Components.Component" />
7. public sealed class AnimationNode : Component
8. {
9. public Material Animation = null;
10. public bool IsLoop = false;
11. private AnimationNode()
12. : base(null)
13. { }
14. public AnimationNode(string Name, Material Source)
15. : base(null)
16. {
17. this.Name = Name;
18. this.Animation = Source;
19. }
20. public override void Destroy()
21. { }
22. }
23. }

#### Core/Components/Animator.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using System;
5. using System.Collections.Generic;
6. using System.Diagnostics;
7. using System.Linq;
8. namespace DKEngine.Core.Components
9. {
10. /// <summary>
11. /// Used for GameObject material animation
12. /// </summary>
13. /// <seealso cref="DKEngine.Core.Components.Behavior" />
14. /// <seealso cref="DKEngine.IAnimated" />
15. public class Animator : Behavior, IAnimated
16. {
17. public TimeSpan CurrentAnimationTime;
18. internal Dictionary<string, AnimationNode> Animations;
19. private AnimationNode \_current;
20. public int NumberOfPlays { get; private set; } = 0;
21. public AnimationNode Current
22. {
23. get { return \_current; }
24. set
25. {
26. if (value != \_current)
27. {
28. \_current = value;
29. Parent.Model = \_current.Animation;
30. NumberOfPlays = 0;
31. CurrentAnimationTime = new TimeSpan(0);
32. }
33. }
34. }
35. public int AnimationState
36. {
37. get
38. {
39. return (int)(CurrentAnimationTime.TotalMilliseconds / Parent.Model.DurationPerFrame % Parent.Model.Frames);
40. }
41. }
42. public Animator(GameObject Parent)
43. : base(Parent)
44. {
45. this.CurrentAnimationTime = new TimeSpan(0);
46. this.Animations = new Dictionary<string, AnimationNode>();
47. this.Name = string.Format("{0}\_{1}", Parent.Name, nameof(Animator));
48. }
49. /// <summary>
50. /// Adds the animation.
51. /// </summary>
52. /// <param name="Name">The animation node name.</param>
53. /// <param name="Source">The source material for animation node.</param>
54. public void AddAnimation(string Name, Material Source)
55. {
56. Animations.Add(Name, new AnimationNode(Name, Source));
57. if (Animations.Count == 1)
58. {
59. Play(Animations.ElementAt(0).Key);
60. }
61. }
62. /// <summary>
63. /// Adds the animation.
64. /// </summary>
65. /// <param name="Name">The animation node name.</param>
66. /// <param name="MaterialKey">The material key to search for material.</param>
67. public void AddAnimation(string Name, string MaterialKey)
68. {
69. Animations.Add(Name, new AnimationNode(Name, Database.GetGameObjectMaterial(MaterialKey)));
70. if (Animations.Count == 1)
71. {
72. Play(Animations.ElementAt(0).Key);
73. }
74. }
75. /// <summary>
76. /// Plays the specified animation name.
77. /// </summary>
78. /// <param name="AnimationName">Name of the animation.</param>
79. public void Play(string AnimationName)
80. {
81. if (AnimationName != Current?.Name)
82. {
83. AnimationNode Result;
84. try
85. {
86. Result = Animations[AnimationName];
87. }
88. catch (Exception e)
89. {
90. Debug.WriteLine("Animation \"{0}\"not found\n{1}", AnimationName, e);
91. return;
92. }
93. Current = Result;
94. }
95. }
96. protected internal override void Update()
97. {
98. if (Parent?.Model?.Frames > 1)
99. {
100. CurrentAnimationTime = CurrentAnimationTime.Add(new TimeSpan(0, 0, 0, 0, (int)(Engine.DeltaTime \* 1000)));
101. if (CurrentAnimationTime.TotalMilliseconds > Parent.Model.Duration)
102. {
103. CurrentAnimationTime = CurrentAnimationTime.Subtract(new TimeSpan(0, 0, 0, 0, Parent.Model.Duration));
104. NumberOfPlays++;
105. }
106. }
107. }
108. protected internal override void Start()
109. { }
110. public override void Destroy()
111. {
112. Engine.UpdateEvent -= UpdateHandle;
113. Parent = null;
114. UpdateHandle = null;
115. }
116. }
117. }

#### Core/Components/Behavior.cs

1. using System;
2. using System.Diagnostics;
3. namespace DKEngine.Core.Components
4. {
5. /// <summary>
6. /// Base class for updated components
7. /// </summary>
8. /// <seealso cref="DKEngine.Core.Components.Component" />
9. public abstract class Behavior : Component
10. {
11. internal Engine.EngineHandler UpdateHandle;
12. public Behavior(GameObject Parent)
13. : base(Parent)
14. {
15. UpdateHandle = new Engine.EngineHandler(Update);
16. }
17. internal sealed override void Init()
18. {
19. try
20. {
21. Engine.LoadingScene.AllBehaviors.Add(this);
22. Engine.LoadingScene.NewlyGeneratedBehaviors.Push(this);
23. }
24. catch (Exception e)
25. {
26. Debug.WriteLine("Loading scene is NULL\n\n{0}", e);
27. }
28. }
29. /// <summary>
30. /// Called before parent object is rendered for first time
31. /// </summary>
32. protected internal abstract void Start();
33. /// <summary>
34. /// Updates each frame once
35. /// </summary>
36. protected internal abstract void Update();
37. }
38. }

#### Core/Components/Camera.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using DKEngine.Core.Ext;
5. using System;
6. using System.Collections.Generic;
7. using System.Drawing;
8. using System.Linq;
9. namespace DKEngine.Core.Components
10. {
11. /// <summary>
12. /// Camera used for rendering
13. /// </summary>
14. /// <seealso cref="DKEngine.Core.Components.Component" />
15. public sealed class Camera : Component
16. {
17. /// <summary>
18. /// Sets the canvas background color
19. /// </summary>
20. public Color BackGround = Color.Black;
21. /// <summary>
22. /// The offset position of camera
23. /// </summary>
24. public Vector3 Position;
25. internal float X { get { return RenderingGUI ? 0 : Parent != null ? Parent.Transform.Position.X + Position.X : Position.X; } }
26. internal float Y { get { return RenderingGUI ? 0 : Parent != null ? Parent.Transform.Position.Y + Position.Y : Position.Y; } }
27. private bool RenderingGUI = false;
28. public Camera()
29. : base(null)
30. {
31. this.Position = new Vector3(0, 0, 0);
32. Engine.LoadingScene.BaseCamera = this;
33. this.Name = string.Format("{0}", nameof(Camera));
34. }
35. public Camera(GameObject Parent)
36. : base(Parent)
37. {
38. Engine.LoadingScene.BaseCamera = this;
39. this.Name = string.Format("{0}\_{1}", Parent.Name, nameof(Camera));
40. }
41. internal void BufferImage(List<GameObject> GameObjectsInView)
42. {
43. BackGroundInit();
44. RenderingGUI = true;
45. List<GameObject> GUI = GameObjectsInView.Where(item => item.IsGUI).ToList();
46. int GUICount = GUI.Count;
47. for (int i = 0; i < GUICount; i++)
48. GameObjectsInView.Remove(GUI[i]);
49. while (GUICount > 0)
50. {
51. float tempHeight = GUI.FindMaxZ();
52. GameObject[] toRender = GUI.Where(item => item.Transform.Position.Z == tempHeight).ToArray();
53. int toRenderCount = toRender.Length;
54. for (int i = toRenderCount - 1; i >= 0; i--)
55. {
56. toRender[i].Render();
57. GUI.Remove(toRender[i]);
58. GUICount--;
59. }
60. }
61. RenderingGUI = false;
62. int TempCount = GameObjectsInView.Count;
63. while (TempCount > 0)
64. {
65. float tempHeight = GameObjectsInView.FindMaxZ();
66. GameObject[] toRender = GameObjectsInView.Where(item => item.Transform.Position.Z == tempHeight).ToArray();
67. int toRenderCount = toRender.Length;
68. for (int i = toRenderCount - 1; i >= 0; i--)
69. {
70. toRender[i].Render();
71. GameObjectsInView.Remove(toRender[i]);
72. TempCount--;
73. }
74. }
75. }
76. private void BackGroundInit()
77. {
78. byte R = BackGround.R;
79. byte G = BackGround.G;
80. byte B = BackGround.B;
81. int imageBufferLenght = Engine.Render.imageBuffer.Length;
82. for (int i = 0; i < imageBufferLenght; i += 3)
83. {
84. Engine.Render.imageBuffer[i + 2] = R;
85. Engine.Render.imageBuffer[i + 1] = G;
86. Engine.Render.imageBuffer[i] = B;
87. }
88. Array.Clear(Engine.Render.imageBufferKey, 0, Engine.Render.imageBufferKey.Length);
89. }
90. public sealed override void Destroy()
91. {
92. if (Engine.BaseCam == this)
93. Engine.BaseCam = null;
94. try
95. {
96. Engine.LoadingScene.AllComponents.Remove(this.Name);
97. }
98. catch
99. { }
100. Parent = null;
101. }
102. }
103. }

#### Core/Components/Collider.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using System;
5. using System.Collections.Generic;
6. using System.Diagnostics;
7. using System.Drawing;
8. using static DKEngine.Core.Components.Transform;
9. namespace DKEngine.Core.Components
10. {
11. /// <summary>
12. /// Collider used for GameObjects
13. /// </summary>
14. /// <seealso cref="DKEngine.Core.Components.Component" />
15. public class Collider : Component
16. {
17. internal event CollisionEnterHandler CollisionEvent;
18. internal delegate void CollisionEnterHandler(Collider m);
19. /// <summary>
20. /// Determines size and position of collider
21. /// </summary>
22. public RectangleF Area = new RectangleF();
23. /// <summary>
24. /// If is TRUE => Triggers OnColliderEnter once another GameObject enter this collider
25. /// </summary>
26. public bool IsTrigger = false;
27. /// <summary>
28. /// Collider is enabled or disabled
29. /// </summary>
30. public bool Enabled = true;
31. private float X { get { return Parent.Transform.Position.X + Area.X; } }
32. private float Y { get { return Parent.Transform.Position.Y + Area.Y; } }
33. private float Width { get { return Parent.Transform.Scale.X \* Area.Width; } }
34. private float Height { get { return Parent.Transform.Scale.Y \* Area.Height; } }
35. private bool \_Right;
36. private bool \_Left;
37. private bool \_Top;
38. private bool \_Bottom;
39. /// <summary>
40. /// Creates new Instance of Collider class
41. /// </summary>
42. /// <param name="Parent">Parent of collider (determines size of collider)</param>
43. internal Collider(GameObject Parent)
44. : base(Parent)
45. {
46. this.Area = new RectangleF(0, 0, Parent.Transform.Dimensions.X, Parent.Transform.Dimensions.Y);
47. this.Name = string.Format("{0}\_{1}", Parent.Name, nameof(Collider));
48. }
49. #if DEBUG
50. /// <summary>
51. /// Returns string containing <b>bool</b> value for each of the directions of this object.
52. /// </summary>
53. /// <returns></returns>
54. public string DebugTestCollision()
55. {
56. return string.Format("Left {0}\nRight {1}\nTop {2}\nDown {3}", Collision(Direction.Left), Collision(Direction.Right), Collision(Direction.Up), Collision(Direction.Down));
57. }
58. #endif
59. /// <summary>
60. /// Collision check in specified direction.
61. /// </summary>
62. /// <param name="direction"></param>
63. /// <returns></returns>
64. public bool Collision(Direction direction)
65. {
66. if (this.IsTrigger || !this.Enabled)
67. return false;
68. if (LastUpdated != Engine.LastUpdated)
69. {
70. \_Right = false;
71. \_Left = false;
72. \_Bottom = false;
73. \_Top = false;
74. int count = Engine.CurrentScene.AllGameObjectsColliders.Count;
75. for (int i = 0; i < count; i++)
76. {
77. Collider tmp = Engine.CurrentScene.AllGameObjectsColliders[i];
78. bool \_L = false;
79. bool \_R = false;
80. bool \_T = false;
81. bool \_B = false;
82. float \_LeftSpan = float.MaxValue;
83. float \_RightSpan = float.MaxValue;
84. float \_BottomSpan = float.MaxValue;
85. float \_TopSpan = float.MaxValue;
86. if (\_L = Left(tmp))
87. {
88. \_LeftSpan = LeftSpan(tmp);
89. }
90. if (\_R = Right(tmp))
91. {
92. \_RightSpan = RightSpan(tmp);
93. }
94. if (\_T = Up(tmp))
95. {
96. \_TopSpan = TopSpan(tmp);
97. }
98. if (\_B = Down(tmp))
99. {
100. \_BottomSpan = BottomSpan(tmp);
101. }
102. if (\_T && \_TopSpan <= \_LeftSpan && \_TopSpan <= \_RightSpan && \_TopSpan <= \_BottomSpan)
103. {
104. \_Top = true;
105. this.Parent.Transform.Position += new Vector3(0, \_TopSpan, 0);
106. continue;
107. }
108. if (\_B && \_BottomSpan <= \_LeftSpan && \_BottomSpan <= \_RightSpan && \_BottomSpan <= \_TopSpan)
109. {
110. \_Bottom = true;
111. this.Parent.Transform.Position += new Vector3(0, -\_BottomSpan, 0);
112. continue;
113. }
114. if (\_L && \_LeftSpan <= \_BottomSpan && \_LeftSpan <= \_TopSpan && \_LeftSpan <= \_RightSpan)
115. {
116. \_Left = true;
117. this.Parent.Transform.Position += new Vector3(\_LeftSpan, 0, 0);
118. continue;
119. }
120. if (\_R && \_RightSpan <= \_BottomSpan && \_RightSpan <= \_TopSpan && \_RightSpan <= \_LeftSpan)
121. {
122. \_Right = true;
123. this.Parent.Transform.Position += new Vector3(-\_RightSpan, 0, 0);
124. continue;
125. }
126. }
127. }
128. switch (direction)
129. {
130. case Direction.Up:
131. return \_Top;
132. case Direction.Left:
133. return \_Left;
134. case Direction.Down:
135. return \_Bottom;
136. case Direction.Right:
137. return \_Right;
138. default:
139. return false;
140. }
141. }
142. internal void TriggerCheck(List<Collider> VisibleObjects)
143. {
144. if (!this.Enabled)
145. return;
146. int VisibleObjectsCount = VisibleObjects.Count;
147. for (int i = 0; i < VisibleObjectsCount; i++)
148. {
149. Collider tmp = VisibleObjects[i];
150. if (!tmp.Enabled)
151. continue;
152. if (Collided(tmp))
153. {
154. CollisionEvent?.Invoke(VisibleObjects[i]);
155. continue;
156. }
157. }
158. }
159. private float LeftSpan(Collider obj)
160. {
161. return obj.X + obj.Width - this.X;
162. }
163. private float TopSpan(Collider obj)
164. {
165. return obj.Y + obj.Height - this.Y;
166. }
167. private float RightSpan(Collider obj)
168. {
169. return this.X + this.Width - obj.X;
170. }
171. private float BottomSpan(Collider obj)
172. {
173. return this.Y + this.Height - obj.Y;
174. }
175. private bool Left(Collider obj)
176. {
177. try
178. {
179. if (!this.Equals(obj) && !obj.IsTrigger && obj.Enabled)
180. return (this.Y < obj.Y + obj.Height && this.Y + this.Height > obj.Y && this.X >= obj.X + obj.Width / 2 && this.X <= obj.X + obj.Width); //(this.Y < obj.Y + obj.Width && this.Y + this.Width > obj.Y && this.X <= obj.X + obj.Width && this.X > obj.X);
181. }
182. catch { }
183. return false;
184. }
185. private bool Right(Collider obj)
186. {
187. try
188. {
189. if (!this.Equals(obj) && !obj.IsTrigger && obj.Enabled)
190. return (this.Y < obj.Y + obj.Height && this.Y + this.Height > obj.Y && this.X + this.Width >= obj.X && this.X + this.Width <= obj.X + obj.Width / 2);//(this.Y < obj.Y + obj.Width && this.Y + this.Width > obj.Y && this.X + this.Width >= obj.X && this.X < X);
191. }
192. catch { }
193. return false;
194. }
195. private bool Up(Collider obj)
196. {
197. try
198. {
199. if (!this.Equals(obj) && !obj.IsTrigger && obj.Enabled)
200. return (this.X < obj.X + obj.Width && this.X + this.Width > obj.X && this.Y <= obj.Y + obj.Height && this.Y >= obj.Y + obj.Height / 2);//(this.X < obj.X + obj.Width && this.X + this.Width > obj.X && this.Y <= obj.Y + obj.Width && this.Y > obj.Y);
201. }
202. catch { }
203. return false;
204. }
205. private bool Down(Collider obj)
206. {
207. try
208. {
209. if (!this.Equals(obj) && !obj.IsTrigger && obj.Enabled)
210. return (this.X < obj.X + obj.Width && this.X + this.Width > obj.X && this.Y + this.Height >= obj.Y && this.Y + this.Height <= obj.Y + obj.Height / 2);//(this.X < obj.X + obj.Width && this.X + this.Width > obj.X && this.Y + this.Width >= obj.Y && this.Y < obj.Y);
211. }
212. catch { }
213. return false;
214. }
215. private bool Collided(Collider obj)
216. {
217. try
218. {
219. if (!this.Equals(obj) && !obj.IsTrigger && obj.Enabled)
220. return (this.X < obj.X + obj.Width && this.X + this.Width > obj.X && this.Y < obj.Y + obj.Height && this.Y + this.Height > obj.Y);
221. }
222. catch { }
223. return false;
224. }
225. public override void Destroy()
226. {
227. try
228. {
229. Engine.CurrentScene.AllGameObjectsColliders.Remove(this);
230. }
231. catch { }
232. try
233. {
234. Engine.CurrentScene.AllComponents.Remove(this.Name);
235. }
236. catch
237. { }
238. if (Parent.Collider == this)
239. Parent.Collider = null;
240. }
241. public void SetCollisionManually(Direction direction)
242. {
243. switch (direction)
244. {
245. case Direction.Up:
246. \_Top = true;
247. break;
248. case Direction.Left:
249. \_Left = true;
250. break;
251. case Direction.Down:
252. \_Bottom = true;
253. break;
254. case Direction.Right:
255. \_Right = true;
256. break;
257. default:
258. break;
259. }
260. }
261. internal sealed override void Init()
262. {
263. try
264. {
265. Engine.LoadingScene.AllGameObjectsColliders.Add(this);
266. }
267. catch (Exception e)
268. {
269. Debug.WriteLine("Loading scene is NULL\n\n{0}", e);
270. }
271. }
272. }
273. }

#### Core/Components/Component.cs

1. using DKEngine.Core.Ext;
2. using DKEngine.Core.UI;
3. using System;
4. using System.Diagnostics;
5. namespace DKEngine.Core.Components
6. {
7. /// <summary>
8. /// Base class for all objects using DKEngine library
9. /// </summary>
10. public abstract class Component
11. {
12. private TimeSpan \_lastUpdated;
13. internal TimeSpan LastUpdated
14. {
15. get
16. {
17. TimeSpan tmp = \_lastUpdated;
18. \_lastUpdated = Engine.LastUpdated;
19. return tmp;
20. }
21. }
22. /// <summary>
23. /// The parent object of this instance
24. /// </summary>
25. public GameObject Parent = null;
26. /// <summary>
27. /// The name of this instance
28. /// </summary>
29. public string Name = "";
30. internal Component(GameObject Parent)
31. {
32. this.Parent = Parent;
33. \_lastUpdated = Engine.LastUpdated;
34. try
35. {
36. Engine.LoadingScene.NewlyGeneratedComponents.Push(this);
37. }
38. catch (Exception e)
39. {
40. Debug.WriteLine("Loading scene is NULL\n\n{0}", e);
41. }
42. }
43. internal void InitInternal()
44. {
45. Init();
46. try
47. {
48. if (this.GetType() != typeof(Letter))
49. {
50. Engine.LoadingScene.AllComponents.AddSafe(this);
51. }
52. }
53. catch (Exception e)
54. {
55. Debug.WriteLine("Loading scene is NULL\n\n{0}", e);
56. }
57. }
58. internal virtual void Init()
59. { }
60. public abstract void Destroy();
61. /// <summary>
62. /// Finds the specified component of specified name.
63. /// </summary>
64. /// <typeparam name="T">Determines type of desired component</typeparam>
65. /// <param name="Name">The name of desired component.</param>
66. /// <returns></returns>
67. public static T Find<T>(string Name) where T : Component
68. {
69. T retValue = null;
70. try
71. {
72. retValue = (T)Engine.LoadingScene.AllComponents[Name];
73. }
74. catch (Exception ex)
75. {
76. Debug.WriteLine("Object not found\n" + ex);
77. }
78. return retValue;
79. }
80. }
81. }

#### Core/Components/Material.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using System;
5. using System.Drawing;
6. using System.Drawing.Imaging;
7. using System.Runtime.InteropServices;
8. namespace DKEngine.Core.Components
9. {
10. /// <summary>
11. /// Low-Memory Material
12. /// </summary>
13. public sealed class Material
14. {
15. /// <summary>
16. /// Source image used as Texture
17. /// </summary>
18. public readonly Bitmap Texture = null;
19. /// <summary>
20. /// Represents scaled length of image in pixels
21. /// </summary>
22. public readonly int Width = 0;
23. /// <summary>
24. /// Represents scaled height of image in pixels
25. /// </summary>
26. public readonly int Height = 0;
27. /// <summary>
28. /// Number of frames
29. /// </summary>
30. public readonly int Frames = 1;
31. /// <summary>
32. /// Total duration of animated image
33. /// </summary>
34. public readonly int Duration = 1;
35. /// <summary>
36. /// Duration between two frames of animation
37. /// </summary>
38. public readonly int DurationPerFrame = 1;
39. /// <summary>
40. /// Returns true if image is animated
41. /// </summary>
42. public readonly bool IsAnimated = false;
43. /// <summary>
44. /// Returns true if image is looped
45. /// </summary>
46. public readonly bool IsLooped = false;
47. private int \_SelectedLayer = -1;
48. private FrameDimension \_FrameDim = null;
49. private BitmapData \_BitmapData = null;
50. private byte[] \_Data = null;
51. private byte \_BytesPerPixel = 0;
52. /// <summary>
53. /// Loads image and creates new material
54. /// </summary>
55. /// <param name="source">Source image</param>
56. public Material(Image source)
57. {
58. if (source != null)
59. {
60. \_FrameDim = new FrameDimension(source.FrameDimensionsList[0]);
61. Frames = source.GetFrameCount(\_FrameDim);
62. Texture = (Bitmap)source;
63. Width = source.Width;
64. Height = source.Height;
65. if (ImageAnimator.CanAnimate(source))
66. {
67. int delay = 0;
68. int this\_delay = 0;
69. int index = 0;
70. for (int i = 0; i < Frames; i++)
71. {
72. this\_delay = BitConverter.ToInt32(source.GetPropertyItem(20736).Value, index) \* 10;
73. delay += (this\_delay < 1 ? 33 : this\_delay);
74. index += 4;
75. }
76. Duration = delay;
77. DurationPerFrame = Duration / Frames;
78. IsAnimated = true;
79. IsLooped = BitConverter.ToInt16(source.GetPropertyItem(20737).Value, 0) != 1;
80. }
81. switch (source.PixelFormat)
82. {
83. case PixelFormat.Format32bppArgb:
84. \_BytesPerPixel = 4;
85. break;
86. case PixelFormat.Format24bppRgb:
87. \_BytesPerPixel = 3;
88. break;
89. default:
90. throw new Exception("Unsupported");
91. }
92. \_Data = new byte[Width \* Height \* \_BytesPerPixel];
93. }
94. }
95. /// <summary>
96. /// Creates new material with given color and scales it by parent's given scales
97. /// </summary>
98. /// <param name="clr">Source color</param>
99. /// <param name="Size">Vector3 used for material size</param>
100. public Material(Color clr, Vector3 Size)
101. {
102. this.Width = (int)(Size.X < 1 ? 1 : Size.X);
103. this.Height = (int)(Size.Y < 1 ? 1 : Size.Y);
104. Frames = 1;
105. \_BytesPerPixel = 4;
106. int size = Width \* Height \* \_BytesPerPixel;
107. \_Data = new byte[size];
108. for (int index = 0; index < size; index += \_BytesPerPixel)
109. {
110. \_Data[index + 3] = clr.A;
111. \_Data[index + 2] = clr.R;
112. \_Data[index + 1] = clr.G;
113. \_Data[index] = clr.B;
114. }
115. unsafe
116. {
117. fixed (byte\* data = \_Data)
118. {
119. using (Bitmap tmp = new Bitmap(Width,
120. Height,
121. Width \* \_BytesPerPixel,
122. PixelFormat.Format32bppArgb,
123. new IntPtr(data)))
124. {
125. Texture = new Bitmap(tmp);
126. }
127. }
128. }
129. \_FrameDim = new FrameDimension(Texture.FrameDimensionsList[0]);
130. }
131. /// <summary>
132. /// Creates new material with given color and scales it by parent's given scales
133. /// </summary>
134. /// <param name="clr">Source color</param>
135. /// <param name="Parent">GameObject used for material size</param>
136. public Material(Color clr, GameObject Parent)
137. : this(clr, Parent.Transform.Dimensions)
138. { }
139. /// <summary>
140. /// Render material into engine image buffer
141. /// </summary>
142. /// <param name="Parent">I3Dimensional for coordiantions</param>
143. public void Render(GameObject Parent, Color? ReColor = null)
144. {
145. int AnimationState = Parent.Animator != null ? Parent.Animator.AnimationState : 0;
146. bool HasShadow = Parent.HasShadow;
147. if (\_SelectedLayer != AnimationState)
148. {
149. if (\_BitmapData != null)
150. Texture.UnlockBits(\_BitmapData);
151. Texture.SelectActiveFrame(\_FrameDim, AnimationState);
152. \_BitmapData = Texture.LockBits(new Rectangle(0, 0, Width, Height), ImageLockMode.ReadOnly, Texture.PixelFormat);
153. Marshal.Copy(\_BitmapData.Scan0, \_Data, 0, \_Data.Length);
154. \_SelectedLayer = AnimationState;
155. }
156. float CamX = Engine.BaseCam != null ? Engine.BaseCam.X : 0;
157. float CamY = Engine.BaseCam != null ? Engine.BaseCam.Y : 0;
158. int x = (int)(Parent.Transform.Position.X - CamX);
159. int y = (int)(Parent.Transform.Position.Y - CamY);
160. float RasteredHeight = this.Height \* Parent.Transform.Scale.Y;
161. float RasteredWidth = this.Width \* Parent.Transform.Scale.X;
162. float NonRasteredWidthRatio = 1 / Parent.Transform.Scale.X;
163. float NonRasteredHeightRatio = 1 / Parent.Transform.Scale.Y;
164. float NonRasteredHeight = 0;
165. float NonRasteredWidth = 0;
166. if (ReColor == null)
167. {
168. for (int row = 0; row < RasteredHeight; row++)
169. {
170. NonRasteredWidth = 0;
171. if (y + row >= Engine.Render.RenderHeight)
172. break;
173. for (int column = 0; column < RasteredWidth; column++)
174. {
175. if (x + column >= Engine.Render.RenderWidth)
176. break;
177. if (IsOnScreen(x + column, y + row))
178. {
179. int offset = (int)(3 \* ((y + row) \* Engine.Render.RenderWidth + (x + column)));
180. int keyOffset = (int)((y + row) \* Engine.Render.RenderWidth + (x + column));
181. int tempColumn = (int)NonRasteredWidth;
182. int tempRow = (int)NonRasteredHeight;
183. int index = \_BytesPerPixel \* (tempRow \* Width + tempColumn);
184. if (Engine.Render.imageBufferKey[keyOffset] != 255 && \_BytesPerPixel == 4 ? \_Data[index + 3] != 0 : true)
185. {
186. Color temp = MixPixel(Engine.Render.imageBufferKey[keyOffset], Engine.Render.imageBuffer[offset + 2], Engine.Render.imageBuffer[offset + 1], Engine.Render.imageBuffer[offset],
187. \_BytesPerPixel == 4 ? \_Data[index + 3] : (byte)255, \_Data[index + 2], \_Data[index + 1], \_Data[index]);
188. Engine.Render.imageBufferKey[keyOffset] = temp.A;
189. Engine.Render.imageBuffer[offset] = temp.B;
190. Engine.Render.imageBuffer[offset + 1] = temp.G;
191. Engine.Render.imageBuffer[offset + 2] = temp.R;
192. }
193. }
194. NonRasteredWidth += NonRasteredWidthRatio;
195. }
196. NonRasteredHeight += NonRasteredHeightRatio;
197. }
198. }
199. else
200. {
201. Color tempColor = (Color)ReColor;
202. for (int row = 0; row < RasteredHeight; row++)
203. {
204. NonRasteredWidth = 0;
205. if (y + row >= Engine.Render.RenderHeight)
206. break;
207. for (int column = 0; column < RasteredWidth; column++)
208. {
209. if (x + column >= Engine.Render.RenderWidth)
210. break;
211. if (IsOnScreen(x + column, y + row))
212. {
213. int offset = (int)(3 \* ((y + row) \* Engine.Render.RenderWidth + (x + column)));
214. int keyOffset = (int)((y + row) \* Engine.Render.RenderWidth + (x + column));
215. int tempColumn = (int)NonRasteredWidth;
216. int tempRow = (int)NonRasteredHeight;
217. int index = \_BytesPerPixel \* (tempRow \* Width + tempColumn);
218. if (Engine.Render.imageBufferKey[keyOffset] != 255 && \_BytesPerPixel == 4 ? \_Data[index + 3] != 0 : true)
219. {
220. Color temp = MixPixel(Color.FromArgb(Engine.Render.imageBufferKey[keyOffset], Engine.Render.imageBuffer[offset + 2], Engine.Render.imageBuffer[offset + 1], Engine.Render.imageBuffer[offset]),
221. tempColor);
222. Engine.Render.imageBufferKey[keyOffset] = temp.A;
223. Engine.Render.imageBuffer[offset] = temp.B;
224. Engine.Render.imageBuffer[offset + 1] = temp.G;
225. Engine.Render.imageBuffer[offset + 2] = temp.R;
226. }
227. }
228. NonRasteredWidth += NonRasteredWidthRatio;
229. }
230. NonRasteredHeight += NonRasteredHeightRatio;
231. }
232. }
233. if (HasShadow)
234. {
235. NonRasteredHeight = 0;
236. NonRasteredWidth = 0;
237. x++;
238. y++;
239. for (int row = 0; row < RasteredHeight; row++)
240. {
241. NonRasteredWidth = 0;
242. if (y + row >= Engine.Render.RenderHeight)
243. break;
244. for (int column = 0; column < RasteredWidth; column++)
245. {
246. if (x + column >= Engine.Render.RenderWidth)
247. break;
248. if (IsOnScreen(x + column, y + row))
249. {
250. int offset = (int)(3 \* ((y + row) \* Engine.Render.RenderWidth + (x + column)));
251. int keyOffset = (int)((y + row) \* Engine.Render.RenderWidth + (x + column));
252. int tempColumn = (int)NonRasteredWidth;
253. int tempRow = (int)NonRasteredHeight;
254. int index = \_BytesPerPixel \* (tempRow \* Width + tempColumn);
255. if (Engine.Render.imageBufferKey[keyOffset] != 255 && \_BytesPerPixel == 4 ? \_Data[index + 3] != 0 : true)
256. {
257. Color temp = MixPixel(Engine.Render.imageBufferKey[keyOffset], Engine.Render.imageBuffer[offset + 2], Engine.Render.imageBuffer[offset + 1], Engine.Render.imageBuffer[offset],
258. (byte)192, (byte)0, (byte)0, (byte)0);
259. Engine.Render.imageBufferKey[keyOffset] = temp.A;
260. Engine.Render.imageBuffer[offset] = temp.B;
261. Engine.Render.imageBuffer[offset + 1] = temp.G;
262. Engine.Render.imageBuffer[offset + 2] = temp.R;
263. }
264. }
265. NonRasteredWidth += NonRasteredWidthRatio;
266. }
267. NonRasteredHeight += NonRasteredHeightRatio;
268. }
269. }
270. }
271. public Color MixPixel(byte topA, byte topR, byte topG, byte topB, byte bottomA, byte bottomR, byte bottomG, byte bottomB)
272. {
273. if (topA == 0)
274. return Color.FromArgb(bottomA, bottomR, bottomG, bottomB);
275. if (bottomA == 0)
276. return Color.FromArgb(topA, topR, topG, topB);
277. float opacityTop = (float)topA / 255;
278. byte newA = (byte)(topA + bottomA >= 255 ? 255 : topA + bottomA);
279. byte A = (byte)(newA - topA);
280. float opacityBottom = (float)A / 255;
281. byte R = (byte)(topR \* opacityTop + bottomR \* opacityBottom);
282. byte G = (byte)(topG \* opacityTop + bottomG \* opacityBottom);
283. byte B = (byte)(topB \* opacityTop + bottomB \* opacityBottom);
284. return Color.FromArgb(newA, R, G, B);
285. }
286. public Color MixPixel(Color top, Color bottom)
287. {
288. if (top.A == 0)
289. return bottom;
290. if (bottom.A == 0)
291. return top;
292. float opacityTop = (float)top.A / 255;
293. byte newA = (byte)(top.A + bottom.A >= 255 ? 255 : top.A + bottom.A);
294. byte A = (byte)(newA - top.A);
295. float opacityBottom = (float)A / 255;
296. byte R = (byte)(top.R \* opacityTop + bottom.R \* opacityBottom);
297. byte G = (byte)(top.G \* opacityTop + bottom.G \* opacityBottom);
298. byte B = (byte)(top.B \* opacityTop + bottom.B \* opacityBottom);
299. return Color.FromArgb(newA, R, G, B);
300. }
301. private bool IsOnScreen(float x, float y)
302. {
303. return x >= 0 && x < Engine.Render.RenderWidth && y >= 0 && y < Engine.Render.RenderHeight;
304. }
305. }
306. }

#### Core/Components/Parabola.cs

1. using System;
2. namespace DKEngine.Core.Components
3. {
4. /\*
5. \* ------------------
6. \* DOES NOT WORK YET
7. \* ------------------
8. \*/
9. [Obsolete]
10. public class Parabola : Behavior
11. {
12. public TimeSpan Time;
13. public float Y;
14. private float \_accumulated = 0f;
15. public float Accumulated { get { return \_accumulated; } }
16. public bool Enabled = false;
17. private float Elapsed = 0f;
18. private float[] ValuesInTime;
19. private int NumberOfSamples;
20. private const float SamplesInSecodnd = 1000;
21. private const float X1 = 0f;
22. public float X2 { get; private set; }
23. public Parabola(GameObject Parent)
24. : base(Parent)
25. {
26. Name = string.Format("{0}\_{1}", Parent.Name, nameof(Parabola));
27. }
28. public override void Destroy()
29. { }
30. protected internal override void Start()
31. {
32. NumberOfSamples = (int)Time.TotalMilliseconds;
33. ValuesInTime = new float[NumberOfSamples];
34. float Duration = (float)Time.TotalSeconds;
35. float lastResult = 0f;
36. for (float i = 0; i < NumberOfSamples; i += 0.1f)
37. {
38. float constant = i / 1000f;
39. float result = ((float)Math.Pow(-constant, 2) - (Duration \* constant)) \* Y;
40. ValuesInTime[(int)i] = result - lastResult;
41. lastResult = result;
42. }
43. }
44. protected internal override void Update()
45. {
46. if (Enabled)
47. {
48. \_accumulated = 0;
49. float MaxTime = 0;
50. float LeftoverTime = (float)((Elapsed + Engine.DeltaTime) \* 1000 - Time.TotalMilliseconds);
51. if (LeftoverTime > 0)
52. {
53. MaxTime = (float)Time.TotalSeconds;
54. Enabled = false;
55. }
56. else
57. {
58. MaxTime = Elapsed + Engine.DeltaTime;
59. }
60. int start = (int)(Elapsed \* 1000);
61. int end = (int)(MaxTime \* 1000);
62. for (int i = start; i < end; i++)
63. {
64. \_accumulated += ValuesInTime[i];
65. }
66. Elapsed += Engine.DeltaTime;
67. }
68. }
69. }
70. }

#### Core/Components/SoundSource.cs

1. using NAudio.Wave;
2. using NAudio.Wave.SampleProviders;
3. using System;
4. using System.Collections.Generic;
5. using System.Linq;
6. namespace DKEngine.Core.Components
7. {
8. internal class SoundPlayer
9. {
10. private readonly DirectSoundOut outputDevice;
11. private readonly MixingSampleProvider mixer;
12. private bool IsAvailable = true;
13. internal SoundPlayer(int sampleRate = 44100, int channelCount = 2)
14. {
15. outputDevice = new DirectSoundOut(40);
16. mixer = new MixingSampleProvider(WaveFormat.CreateIeeeFloatWaveFormat(sampleRate, channelCount))
17. {
18. ReadFully = true
19. };
20. outputDevice.Init(mixer);
21. outputDevice.Play();
22. }
23. private ISampleProvider ConvertToRightChannelCount(CachedSoundSampleProvider input)
24. {
25. if (input.WaveFormat.Channels == mixer.WaveFormat.Channels)
26. {
27. input.cachedSound.\_SampleProvider = input;
28. return input.cachedSound.\_SampleProvider;
29. }
30. if (input.WaveFormat.Channels == 1 && mixer.WaveFormat.Channels == 2)
31. {
32. input.cachedSound.\_SampleProvider = new MonoToStereoSampleProvider(input);
33. return input.cachedSound.\_SampleProvider;
34. }
35. throw new NotImplementedException("Not yet implemented this channel count conversion");
36. }
37. public void PlaySound(Sound sound)
38. {
39. if (IsAvailable)
40. {
41. try
42. {
43. sound.\_SampleProvider = ConvertToRightChannelCount(new CachedSoundSampleProvider(sound));
44. AddMixerInput(sound.\_SampleProvider);
45. }
46. catch
47. {
48. IsAvailable = false;
49. }
50. }
51. }
52. public void StopSound(Sound sound)
53. {
54. if (IsAvailable)
55. {
56. try
57. {
58. RemoveMixerInput(sound.\_SampleProvider);
59. }
60. catch
61. { }
62. }
63. }
64. private void AddMixerInput(ISampleProvider input)
65. {
66. mixer.AddMixerInput(input);
67. }
68. private void RemoveMixerInput(ISampleProvider input)
69. {
70. mixer.RemoveMixerInput(input);
71. }
72. public void Dispose()
73. {
74. outputDevice.Dispose();
75. }
76. }
77. /// <summary>
78. /// SoundSource component used for sound effects
79. /// </summary>
80. /// <seealso cref="DKEngine.Core.Components.Component" />
81. public class SoundSource : Component
82. {
83. private bool IsAvailable = true;
84. public SoundSource(GameObject Parent)
85. : base(Parent)
86. {
87. this.Name = string.Format("{0}\_{1}", Parent.Name, nameof(SoundSource));
88. }
89. public void PlaySound(Sound sound)
90. {
91. if (Engine.Sound.IsSoundEnabled)
92. {
93. if (IsAvailable)
94. {
95. try
96. {
97. Engine.Sound.Instance.PlaySound(sound);
98. }
99. catch
100. {
101. IsAvailable = false;
102. }
103. }
104. }
105. }
106. public void StopSound(Sound sound)
107. {
108. if (Engine.Sound.IsSoundEnabled)
109. {
110. if (IsAvailable)
111. {
112. try
113. {
114. Engine.Sound.Instance.StopSound(sound);
115. }
116. catch
117. { }
118. }
119. }
120. }
121. public override void Destroy()
122. {
123. try
124. {
125. Engine.LoadingScene.AllComponents.Remove(this.Name);
126. }
127. catch
128. { }
129. this.Parent = null;
130. }
131. }
132. internal class CachedSoundSampleProvider : ISampleProvider
133. {
134. public Sound cachedSound;
135. private long position;
136. public CachedSoundSampleProvider(Sound cachedSound)
137. {
138. this.cachedSound = cachedSound;
139. }
140. public int Read(float[] buffer, int offset, int count)
141. {
142. var availableSamples = cachedSound.AudioData.Length - position;
143. var samplesToCopy = Math.Min(availableSamples, count);
144. Array.Copy(cachedSound.AudioData, position, buffer, offset, samplesToCopy);
145. position += samplesToCopy;
146. return (int)samplesToCopy;
147. }
148. public WaveFormat WaveFormat { get { return cachedSound.WaveFormat; } }
149. }
150. /// <summary>
151. /// Class holding specified audio file
152. /// </summary>
153. public class Sound
154. {
155. public float[] AudioData { get; private set; }
156. public WaveFormat WaveFormat { get; private set; }
157. public AudioFileReader FileReader { get; private set; }
158. internal ISampleProvider \_SampleProvider { get; set; }
159. public Sound(string audioFileName)
160. {
161. using (FileReader = new AudioFileReader(audioFileName))
162. {
163. FileReader.Volume = Engine.Sound.SoundVolume;
164. // TODO: could add resampling in here if required
165. WaveFormat = FileReader.WaveFormat;
166. var wholeFile = new List<float>((int)(FileReader.Length / 4));
167. var readBuffer = new float[FileReader.WaveFormat.SampleRate \* FileReader.WaveFormat.Channels];
168. int samplesRead;
169. while ((samplesRead = FileReader.Read(readBuffer, 0, readBuffer.Length)) > 0)
170. {
171. wholeFile.AddRange(readBuffer.Take(samplesRead));
172. }
173. AudioData = wholeFile.ToArray();
174. }
175. }
176. }
177. }

#### Core/Components/Transform.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. namespace DKEngine.Core.Components
5. {
6. /// <summary>
7. /// Transformation class holding information about position, scale and sizes of GameObject
8. /// </summary>
9. /// <seealso cref="DKEngine.Core.Components.Component" />
10. public sealed class Transform : Component
11. {
12. private Vector3 \_Dimensions;
13. private Vector3 \_Position;
14. private Vector3 \_Scale;
15. public Vector3 Dimensions
16. {
17. get { return \_Dimensions; }
18. set
19. {
20. Vector3 tmp = value - \_Dimensions;
21. \_Dimensions = value;
22. \_ScaledDimensions = \_Dimensions \* \_Scale;
23. int childCount = Parent.Child.Count;
24. for (int i = 0; i < childCount; i++)
25. Parent.Child[i].Transform.Dimensions += tmp;
26. }
27. }
28. public Vector3 Position
29. {
30. get { return \_Position; }
31. set
32. {
33. Vector3 tmp = value - \_Position;
34. \_Position = value;
35. int childCount = Parent.Child.Count;
36. for (int i = 0; i < childCount; i++)
37. Parent.Child[i].Transform.Position += tmp;
38. }
39. }
40. public Vector3 Scale
41. {
42. get { return \_Scale; }
43. set
44. {
45. Vector3 tmp = value / \_Scale;
46. \_Scale = value;
47. \_ScaledDimensions = \_Dimensions \* \_Scale;
48. int childCount = Parent.Child.Count;
49. for (int i = 0; i < childCount; i++)
50. Parent.Child[i].Transform.Scale \*= tmp;
51. }
52. }
53. internal Vector3 \_ScaledDimensions;
54. public Transform(GameObject Parent)
55. : base(Parent)
56. {
57. \_Position = new Vector3();
58. \_Dimensions = new Vector3();
59. \_Scale = new Vector3();
60. \_ScaledDimensions = new Vector3();
61. }
62. public override void Destroy()
63. {
64. try
65. {
66. Engine.LoadingScene.AllComponents.Remove(this.Name);
67. }
68. catch { }
69. }
70. /// <summary>
71. /// Possible directions
72. /// </summary>
73. public enum Direction
74. {
75. Up,
76. Left,
77. Down,
78. Right
79. }
80. }
81. }

#### Core/Components/Vector3.cs

1. namespace DKEngine.Core.Components
2. {
3. #pragma warning disable CS0660 // Type defines operator == or operator != but does not override Object.Equals(object o)
4. #pragma warning disable CS0661 // Type defines operator == or operator != but does not override Object.GetHashCode()
5. /// <summary>
6. /// Three-dimensional vector
7. /// </summary>
8. public struct Vector3
9. #pragma warning restore CS0661 // Type defines operator == or operator != but does not override Object.GetHashCode()
10. #pragma warning restore CS0660 // Type defines operator == or operator != but does not override Object.Equals(object o)
11. {
12. /// <summary>
13. /// The X vector
14. /// </summary>
15. public float X;
16. /// <summary>
17. /// The Y vector
18. /// </summary>
19. public float Y;
20. /// <summary>
21. /// The Z vector
22. /// </summary>
23. public float Z;
24. public Vector3(float X, float Y, float Z)
25. {
26. this.X = X;
27. this.Y = Y;
28. this.Z = Z;
29. }
30. public static Vector3 operator -(Vector3 left, Vector3 right)
31. {
32. return new Vector3(left.X - right.X, left.Y - right.Y, left.Z - right.Z);
33. }
34. public static Vector3 operator -(Vector3 left, float right)
35. {
36. return new Vector3(left.X - right, left.Y - right, left.Z - right);
37. }
38. public static Vector3 operator +(Vector3 left, Vector3 right)
39. {
40. return new Vector3(left.X + right.X, left.Y + right.Y, left.Z + right.Z);
41. }
42. public static Vector3 operator +(Vector3 left, float right)
43. {
44. return new Vector3(left.X + right, left.Y + right, left.Z + right);
45. }
46. public static Vector3 operator \*(Vector3 left, Vector3 right)
47. {
48. return new Vector3(left.X \* right.X, left.Y \* right.Y, left.Z \* right.Z);
49. }
50. public static Vector3 operator \*(Vector3 left, float right)
51. {
52. return new Vector3(left.X \* right, left.Y \* right, left.Z \* right);
53. }
54. public static Vector3 operator /(Vector3 left, Vector3 right)
55. {
56. return new Vector3(left.X / right.X, left.Y / right.Y, left.Z / right.Z);
57. }
58. public static Vector3 operator /(Vector3 left, float right)
59. {
60. return new Vector3(left.X / right, left.Y / right, left.Z / right);
61. }
62. public static bool operator ==(Vector3 left, Vector3 right)
63. {
64. return left.X == right.X && left.Y == right.Y && left.Z == right.Z;
65. }
66. public static bool operator !=(Vector3 left, Vector3 right)
67. {
68. return left.X != right.X || left.Y != right.Y || left.Z != right.Z;
69. }
70. public Vector3 Add(Vector3 Value)
71. {
72. return this + Value;
73. }
74. public Vector3 Add(float X, float Y, float Z)
75. {
76. return this + new Vector3(X, Y, Z);
77. }
78. public Vector3 Add(float Value)
79. {
80. return this + Value;
81. }
82. public Vector3 Decrease(Vector3 Value)
83. {
84. return this - Value;
85. }
86. public Vector3 Decrease(float X, float Y, float Z)
87. {
88. return this - new Vector3(X, Y, Z);
89. }
90. public Vector3 Decrease(float Value)
91. {
92. return this - Value;
93. }
94. public Vector3 Multiply(Vector3 Value)
95. {
96. return this \* Value;
97. }
98. public Vector3 Multiply(float X, float Y, float Z)
99. {
100. return this \* new Vector3(X, Y, Z);
101. }
102. public Vector3 Multiply(float Value)
103. {
104. return this \* Value;
105. }
106. public Vector3 Divide(Vector3 Value)
107. {
108. return this / Value;
109. }
110. public Vector3 Divide(float X, float Y, float Z)
111. {
112. return this / new Vector3(X, Y, Z);
113. }
114. public Vector3 Divide(float Value)
115. {
116. return this / Value;
117. }
118. private static Vector3 \_zero = new Vector3(0, 0, 0);
119. public static Vector3 Zero
120. {
121. get { return \_zero; }
122. }
123. }
124. }

#### Core/Scripts/TextControlScript.cs

1. using DKEngine.Core.Components;
2. using DKEngine.Core.UI;
3. using System.Collections.Generic;
4. using static DKEngine.Core.UI.Text;
5. namespace DKEngine.Core.Scripts
6. {
7. internal sealed class TextControlScript : Script
8. {
9. internal TextBlock \_Parent { get { return (TextBlock)Parent; } }
10. public TextControlScript(TextBlock Parent)
11. : base(Parent)
12. { }
13. protected internal override void Start()
14. {
15. if (\_Parent.Text.Length > 0)
16. {
17. Text();
18. }
19. }
20. protected internal override void Update()
21. {
22. if (\_Parent != null)
23. {
24. if (\_Parent.\_changed)
25. {
26. Text();
27. }
28. }
29. }
30. private void Text()
31. {
32. int \_textCount = \_Parent.\_text.Count;
33. for (int i = \_textCount - 1; i >= 0; i--)
34. \_Parent.\_text[i].Destroy();
35. List<Letter> retValue = new List<Letter>();
36. List<List<Letter>> textAligned = new List<List<Letter>>() { new List<Letter>() };
37. float Xoffset = 0;
38. float Yoffset = 0;
39. int rows = 0;
40. if (\_Parent.Transform.Dimensions.X > 0)
41. {
42. for (int i = 0; i < \_Parent.\_textStr.Length; i++)
43. {
44. if (\_Parent.\_textStr[i] == ' ')
45. {
46. Xoffset += 3 \* \_Parent.Transform.Scale.X \* \_Parent.FontSize;
47. }
48. else
49. {
50. if (\_Parent.\_textStr[i] == '\r' || \_Parent.\_textStr[i] == '\n')
51. {
52. Xoffset = 0;
53. Yoffset += 6 \* \_Parent.Transform.Scale.Y \* \_Parent.FontSize;
54. rows++;
55. textAligned.Add(new List<Letter>());
56. continue;
57. }
58. Material newLetterMaterial = Database.GetLetter(\_Parent.\_textStr[i]);
59. if (Xoffset + newLetterMaterial.Width \* \_Parent.FontSize > \_Parent.Transform.Dimensions.X)
60. {
61. Xoffset = 0;
62. Yoffset += 6 \* \_Parent.Transform.Scale.Y \* \_Parent.FontSize;
63. rows++;
64. textAligned.Add(new List<Letter>());
65. }
66. Letter l = new Letter(\_Parent);
67. l.Transform.Position += new Vector3(Xoffset, Yoffset, \_Parent.Transform.Position.Z);
68. l.Foreground = \_Parent.Foreground;
69. l.Model = newLetterMaterial;
70. l.Transform.Scale \*= \_Parent.FontSize;
71. l.Name = \_Parent.\_textStr[i].ToString();
72. l.HasShadow = \_Parent.TextShadow;
73. textAligned[rows].Add(l);
74. Xoffset += (l.Transform.Dimensions.X + 1) \* l.Transform.Scale.X;
75. }
76. }
77. }
78. int textAlignedCount = textAligned.Count;
79. float maxHeight = textAlignedCount \* 6 \* \_Parent.FontSize \* \_Parent.Transform.Scale.Y;
80. float startY = 0;
81. switch (\_Parent.\_TVA)
82. {
83. case VerticalAlignment.Top:
84. startY = 0;
85. break;
86. case VerticalAlignment.Center:
87. startY = (\_Parent.Transform.Dimensions.Y \* \_Parent.Transform.Scale.Y \* \_Parent.FontSize - maxHeight) / 2;
88. break;
89. case VerticalAlignment.Bottom:
90. startY = \_Parent.Transform.Dimensions.Y \* \_Parent.Transform.Scale.Y \* \_Parent.FontSize - maxHeight;
91. break;
92. default:
93. break;
94. }
95. for (int i = 0; i < textAlignedCount; i++)
96. {
97. float maxWidth = 0;
98. int textAlignedRowCount = textAligned[i].Count;
99. if (textAlignedRowCount > 0)
100. maxWidth = textAligned[i][textAlignedRowCount - 1].Model.Width \* \_Parent.Transform.Scale.X \* \_Parent.FontSize + textAligned[i][textAlignedRowCount - 1].Transform.Position.X - textAligned[i][0].Transform.Position.X;
101. if (maxWidth != 0)
102. {
103. float startX = 0;
104. switch (\_Parent.\_THA)
105. {
106. case HorizontalAlignment.Left:
107. startX = 0;
108. break;
109. case HorizontalAlignment.Center:
110. startX = (\_Parent.Transform.Dimensions.X \* \_Parent.Transform.Scale.X - maxWidth) / 2;
111. break;
112. case HorizontalAlignment.Right:
113. startX = \_Parent.Transform.Dimensions.X \* \_Parent.Transform.Scale.X - maxWidth;
114. break;
115. }
116. for (int j = 0; j < textAlignedRowCount; j++)//foreach (Letter letter in row)
117. {
118. if (startX != 0 || startY != 0)
119. textAligned[i][j].Transform.Position += new Vector3(startX, startY, 0);
120. retValue.Add(textAligned[i][j]);
121. }
122. }
123. }
124. \_Parent.\_text = retValue;
125. \_Parent.\_changed = false;
126. }
127. protected internal override void OnColliderEnter(Collider e)
128. { }
129. }
130. }

#### Core/SystemExt/Extensions.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using DKEngine.Core.Components;
5. using System.Collections.Generic;
6. using System.Linq;
7. namespace DKEngine.Core.Ext
8. {
9. public static class Extensions
10. {
11. public static void AddSafe<DataValue>(this Dictionary<string, DataValue> Destination, Component Target)
12. where DataValue : Component
13. {
14. string Key = Target.Name;
15. if (Engine.LoadingScene.ComponentCount.ContainsKey(Target.Name))
16. {
17. Target.Name = string.Format("{0}\_(Copy {1})", Key, Engine.LoadingScene.ComponentCount[Target.Name]++);
18. Key = Target.Name;
19. }
20. else
21. {
22. Engine.LoadingScene.ComponentCount.Add(Key, 1);
23. }
24. Destination.Add(Key, Target as DataValue);
25. }
26. public static void AddAll<T>(this List<T> list, params T[] stuff)
27. {
28. foreach (var item in stuff)
29. {
30. list.Add(item);
31. }
32. }
33. public static float FindMaxZ(this List<GameObject> list)
34. {
35. return list.Max(obj => obj.Transform.Position.Z);
36. }
37. public static List<GameObject> GetGameObjectsInView(this IEnumerable<GameObject> list)
38. {
39. return list.Where(obj => obj.IsInView).ToList();
40. }
41. }
42. }

#### Core/SystemExt/WindowControl.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using System;
5. using System.IO;
6. using System.Runtime.InteropServices;
7. using System.Timers;
8. namespace DKEngine.Core.Ext
9. {
10. /// <summary>
11. /// DKEngine window controller
12. /// </summary>
13. public static class WindowControl
14. {
15. [StructLayout(LayoutKind.Sequential)]
16. private struct COORD
17. {
18. public short X;
19. public short Y;
20. public COORD(short x, short y)
21. {
22. this.X = x;
23. this.Y = y;
24. }
25. }
26. [DllImport("kernel32.dll")]
27. private static extern IntPtr GetStdHandle(int handle);
28. [DllImport("kernel32.dll", SetLastError = true)]
29. private static extern bool SetConsoleDisplayMode(IntPtr ConsoleOutput, uint Flags, out COORD NewScreenBufferDimensions);
30. [DllImport("user32.dll")]
31. public static extern bool ShowWindow(IntPtr hWnd, int cmdShow);
32. private static readonly IntPtr hConsole = GetStdHandle(-11);
33. private static readonly IntPtr INVALID\_HANDLE\_VALUE = new IntPtr(-1);
34. private static COORD xy = new COORD(100, 100);
35. private static bool ConsoleStateChangeAvailable = true;
36. internal static void WindowInit()
37. {
38. Console.CursorVisible = false;
39. Console.SetOut(TextWriter.Null);
40. Console.SetIn(TextReader.Null);
41. Console.BufferHeight = Console.LargestWindowHeight;
42. Console.BufferWidth = Console.LargestWindowWidth;
43. Console.ForegroundColor = ConsoleColor.White;
44. Console.BackgroundColor = ConsoleColor.Black;
45. Console.Clear();
46. WindowSizeChecker(null, null);
47. Timer windowChecker = new Timer()
48. {
49. AutoReset = true,
50. Enabled = true,
51. Interval = 1000f
52. };
53. windowChecker.Elapsed += WindowSizeChecker;
54. windowChecker.Start();
55. }
56. private static void WindowSizeChecker(object sender, ElapsedEventArgs e)
57. {
58. if (Console.WindowHeight != Console.LargestWindowHeight || Console.WindowWidth != Console.LargestWindowWidth)
59. {
60. if (ConsoleStateChangeAvailable)
61. {
62. if (!SetConsoleDisplayMode(hConsole, 1, out xy))
63. {
64. ConsoleStateChangeAvailable = false;
65. }
66. }
67. Console.CursorVisible = false;
68. }
69. }
70. }
71. }

#### Core/UI/Letter.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. namespace DKEngine.Core.UI
5. {
6. internal sealed class Letter : GameObject
7. {
8. private Letter()
9. { }
10. public Letter(TextBlock Parent)
11. : base(Parent)
12. { }
13. public override void Destroy()
14. {
15. try
16. {
17. if (Engine.LoadingScene.NewlyGeneratedComponents.Contains(this))
18. {
19. Engine.LoadingScene.NewlyGeneratedComponents.Pop();
20. }
21. }
22. catch
23. { }
24. try
25. {
26. Engine.RenderObjects.Remove(this);
27. }
28. catch { }
29. Parent?.Child.Remove(this);
30. Animator?.Destroy();
31. Parent = null;
32. Animator = null;
33. Model = null;
34. }
35. }
36. }

#### Core/UI/Text.cs

1. namespace DKEngine.Core.UI
2. {
3. public static class Text
4. {
5. public enum HorizontalAlignment
6. {
7. Left,
8. Center,
9. Right
10. };
11. public enum VerticalAlignment
12. {
13. Top,
14. Center,
15. Bottom
16. };
17. public enum InputType
18. {
19. All,
20. AlphaNumerical,
21. Alpha,
22. Numerical
23. };
24. }
25. }

#### Core/UI/TextBlock.cs

1. /\*
2. \* (C) 2017 David Knieradl
3. \*/
4. using DKEngine.Core.Components;
5. using DKEngine.Core.Scripts;
6. using System.Collections.Generic;
7. using System.Drawing;
8. using static DKEngine.Core.UI.Text;
9. namespace DKEngine.Core.UI
10. {
11. public class TextBlock : GameObject, IText
12. {
13. public virtual string Text
14. {
15. get
16. {
17. return \_textStr ?? throw new System.Exception("WTF PROC KDY KDE A JAK");
18. }
19. set
20. {
21. if (value != \_textStr)
22. {
23. \_textStr = value ?? throw new System.Exception("WTF PROC KDY KDE A JAK");
24. \_changed = true;
25. }
26. }
27. }
28. public Color? Background
29. {
30. get { return \_bg; }
31. set
32. {
33. \_bg = value;
34. if (value != null)
35. Model = new Material((Color)value, this);
36. }
37. }
38. public float FontSize
39. {
40. get { return \_FontSize; }
41. set
42. {
43. if (value <= 0)
44. {
45. \_FontSize = 0.01f;
46. \_changed = true;
47. }
48. else
49. {
50. \_FontSize = value;
51. \_changed = true;
52. }
53. }
54. }
55. public HorizontalAlignment HAlignment
56. {
57. set
58. {
59. \_HA = value;
60. //if (\_IsGUI)
61. //{
62. this.Transform.Position -= new Vector3(horiOffset, 0, 0);
63. switch (value)
64. {
65. case HorizontalAlignment.Left:
66. horiOffset = 0;
67. break;
68. case HorizontalAlignment.Center:
69. horiOffset = (Engine.Render.RenderWidth - this.Transform.\_ScaledDimensions.X) / 2;
70. break;
71. case HorizontalAlignment.Right:
72. horiOffset = Engine.Render.RenderWidth - this.Transform.\_ScaledDimensions.X;
73. break;
74. }
75. this.Transform.Position += new Vector3(horiOffset, 0, 0);
76. //}
77. //\_changed = true;
78. }
79. }
80. public VerticalAlignment VAlignment
81. {
82. set
83. {
84. \_VA = value;
85. //if (\_IsGUI)
86. //{
87. this.Transform.Position -= new Vector3(0, vertOffset, 0);
88. switch (value)
89. {
90. case VerticalAlignment.Top:
91. vertOffset = 0;
92. break;
93. case VerticalAlignment.Center:
94. vertOffset = (Engine.Render.RenderHeight - this.Transform.\_ScaledDimensions.Y) / 2;
95. break;
96. case VerticalAlignment.Bottom:
97. vertOffset = Engine.Render.RenderHeight - this.Transform.\_ScaledDimensions.Y;
98. break;
99. }
100. this.Transform.Position += new Vector3(0, vertOffset, 0);
101. //}
102. //\_changed = true;
103. }
104. }
105. public HorizontalAlignment TextHAlignment
106. {
107. set
108. {
109. \_THA = value;
110. \_changed = true;
111. }
112. }
113. public VerticalAlignment TextVAlignment
114. {
115. set
116. {
117. \_TVA = value;
118. \_changed = true;
119. }
120. }
121. public bool TextShadow = false;
122. internal List<Letter> \_text = new List<Letter>();
123. internal float \_FontSize = 1;
124. internal Color? \_bg;
125. internal string \_textStr = "";
126. internal HorizontalAlignment \_HA = HorizontalAlignment.Left;
127. internal VerticalAlignment \_VA = VerticalAlignment.Top;
128. internal HorizontalAlignment \_THA = HorizontalAlignment.Left;
129. internal VerticalAlignment \_TVA = VerticalAlignment.Top;
130. internal float vertOffset = 0;
131. internal float horiOffset = 0;
132. internal bool \_changed = false;
133. public TextBlock()
134. : base()
135. { }
136. public TextBlock(GameObject Parent)
137. : base(Parent)
138. { }
139. protected override void Initialize()
140. {
141. this.VAlignment = \_VA;
142. this.HAlignment = \_HA;
143. this.InitNewScript<TextControlScript>();
144. }
145. internal override void Render()
146. { Model?.Render(this, \_bg); }
147. public override void Destroy()
148. {
149. base.Destroy();
150. }
151. }
152. }

#### Core/Database.cs

1. using DKEngine.Core.Components;
2. using DKEngine.Properties;
3. using System;
4. using System.Collections;
5. using System.Collections.Generic;
6. using System.Diagnostics;
7. using System.Drawing;
8. using System.IO;
9. using System.Linq;
10. using System.Resources;
11. namespace DKEngine.Core
12. {
13. /// <summary>
14. /// DKEngine library database holding all loaded materials, scenes, etc.
15. /// </summary>
16. public static class Database
17. {
18. private enum Font
19. {
20. Num0,
21. Num1,
22. Num2,
23. Num3,
24. Num4,
25. Num5,
26. Num6,
27. Num7,
28. Num8,
29. Num9,
30. A,
31. AngleBracketLeft,
32. AngleBracketRight,
33. ArrowToLeft,
34. ArrowToRight,
35. ArrowToTop,
36. B,
37. Backslash,
38. BraceLeft,
39. BraceRight,
40. BracketLeft,
41. BracketRight,
42. C,
43. Colon,
44. Comma,
45. D,
46. Dot,
47. E,
48. Equals,
49. ExclamationMark,
50. F,
51. G,
52. H,
53. Hashtag,
54. I,
55. J,
56. K,
57. L,
58. M,
59. Minus,
60. N,
61. O,
62. P,
63. Percent,
64. Q,
65. QuestionMark,
66. QuotationMarks,
67. R,
68. S,
69. Semicolon,
70. Slash,
71. StarLarge,
72. StarSmall,
73. T,
74. U,
75. Underscore,
76. V,
77. W,
78. X,
79. Y,
80. Z,
81. NumberOfTypes
82. };
83. private static Dictionary<char, Font> font = new Dictionary<char, Font>()
84. {
85. { '0' , Font.Num0 },
86. { '1' , Font.Num1 },
87. { '2' , Font.Num2 },
88. { '3' , Font.Num3 },
89. { '4' , Font.Num4 },
90. { '5' , Font.Num5 },
91. { '6' , Font.Num6 },
92. { '7' , Font.Num7 },
93. { '8' , Font.Num8 },
94. { '9' , Font.Num9 },
95. { 'A' , Font.A },
96. { 'B' , Font.B },
97. { 'C' , Font.C },
98. { 'D' , Font.D },
99. { 'E' , Font.E },
100. { 'F' , Font.F },
101. { 'G' , Font.G },
102. { 'H' , Font.H },
103. { 'I' , Font.I },
104. { 'J' , Font.J },
105. { 'K' , Font.K },
106. { 'L' , Font.L },
107. { 'M' , Font.M },
108. { 'N' , Font.N },
109. { 'O' , Font.O },
110. { 'P' , Font.P },
111. { 'Q' , Font.Q },
112. { 'R' , Font.R },
113. { 'S' , Font.S },
114. { 'T' , Font.T },
115. { 'U' , Font.U },
116. { 'V' , Font.V },
117. { 'W' , Font.W },
118. { 'X' , Font.X },
119. { 'Y' , Font.Y },
120. { 'Z' , Font.Z },
121. { '-' , Font.Minus },
122. { '?' , Font.QuestionMark },
123. { '!' , Font.ExclamationMark },
124. { '.' , Font.Dot },
125. { ':' , Font.Colon },
126. { ',' , Font.Comma },
127. { '[' , Font.AngleBracketLeft },
128. { ']' , Font.AngleBracketRight },
129. { '>' , Font.ArrowToRight },
130. { '<' , Font.ArrowToLeft },
131. { '^' , Font.ArrowToTop },
132. { '{' , Font.BraceLeft },
133. { '}' , Font.BraceRight },
134. { '(' , Font.BracketLeft },
135. { ')' , Font.BracketRight },
136. { '=' , Font.Equals },
137. { '#' , Font.Hashtag },
138. { '%' , Font.Percent },
139. { '"' , Font.QuotationMarks },
140. { ';' , Font.Semicolon },
141. { '☼' , Font.StarLarge },
142. { '\*' , Font.StarSmall },
143. { '\_' , Font.Underscore },
144. { '/' , Font.Slash },
145. { '\\' , Font.Backslash }
146. };
147. private static List<Material> letterMaterial = new List<Material>();
148. private static void CreateLetterReferences()
149. {
150. using (BinaryReader br = new BinaryReader(new MemoryStream(Resources.FontFile)))
151. {
152. int lenght = br.ReadInt32();
153. for (int index = 0; index < lenght; index++)
154. {
155. byte[] byteArray = br.ReadBytes(br.ReadInt32());
156. using (MemoryStream ms = new MemoryStream(byteArray))
157. {
158. letterMaterial.Add(new Material((Bitmap)Image.FromStream(ms)));
159. }
160. }
161. }
162. }
163. private static Dictionary<string, Material> CachedMaterials = new Dictionary<string, Material>();
164. private static Dictionary<string, Scene> CachedScenes = new Dictionary<string, Scene>();
165. internal static void InitDatabase()
166. {
167. AddNewGameObjectMaterial("border", new Material(Resources.border));
168. AddNewGameObjectMaterial("splashScreen", new Material(Resources.DKEngine\_splash2));
169. CreateLetterReferences();
170. }
171. internal static Scene GetScene(string Key)
172. {
173. Scene retValue = null;
174. try
175. {
176. retValue = CachedScenes[Key];
177. }
178. catch
179. { }
180. return retValue;
181. }
182. public static Material GetLetter(this char ch)
183. {
184. Material retValue = null;
185. try
186. {
187. retValue = letterMaterial[(int)font[Char.ToUpper(ch)]];
188. }
189. catch
190. {
191. retValue = letterMaterial[(int)font['?']];
192. }
193. return retValue;
194. }
195. public static void AddNewGameObjectMaterial(string ObjectName, Material Object)
196. {
197. try
198. {
199. if (Object != null)
200. {
201. CachedMaterials.Add(ObjectName, Object);
202. }
203. else
204. throw new Exception("Material is null\n" + Object.ToString());
205. }
206. catch (Exception e)
207. {
208. Debug.WriteLine("Object not found\n" + e);
209. }
210. }
211. public static Material GetGameObjectMaterial(string Key)
212. {
213. Material retValue = null;
214. try
215. {
216. retValue = CachedMaterials[Key];
217. }
218. catch (Exception e)
219. {
220. Debug.WriteLine("Object not found\n" + e);
221. }
222. return retValue;
223. }
224. public static Material GetGameObjectMaterial(int Position)
225. {
226. Material retValue = null;
227. try
228. {
229. retValue = CachedMaterials.ElementAtOrDefault(Position).Value;
230. }
231. catch (Exception e)
232. {
233. Debug.WriteLine("Object not found\n" + e);
234. }
235. return retValue;
236. }
237. public static string GetMaterialDatabaseKey(int Position)
238. {
239. return CachedMaterials.ElementAtOrDefault(Position).Key; //.FirstOrDefault(x => x.Value == Position).Key;
240. }
241. public static void LoadResources(ResourceSet source)
242. {
243. foreach (DictionaryEntry entry in source)
244. {
245. if (entry.Value is Image)
246. {
247. AddNewGameObjectMaterial((string)entry.Key, new Material((Bitmap)entry.Value));
248. }
249. }
250. }
251. internal static void RewriteWorld(string Name, object[] argsPreLoad = null)
252. {
253. try
254. {
255. Engine.LoadingScene = CachedScenes[Name];
256. object[] preArgs = argsPreLoad ?? Engine.LoadingScene.argsPreLoad;
257. object[] postArgs = Engine.LoadingScene.argsPostLoad;
258. var list = Engine.LoadingScene.AllComponents.ToList();
259. for (int i = 0; i < list.Count; i++)
260. {
261. list[0].Value.Destroy();
262. list.RemoveAt(0);
263. list = Engine.LoadingScene.AllComponents.ToList();
264. }
265. for (int i = 0; i < Engine.LoadingScene.AllBehaviors.Count; i++)
266. {
267. Engine.LoadingScene.AllBehaviors[0].Destroy();
268. }
269. Engine.LoadingScene = (Scene)Activator.CreateInstance(Engine.LoadingScene.GetType());
270. Engine.LoadingScene.argsPreLoad = preArgs;
271. Engine.LoadingScene.argsPostLoad = postArgs;
272. Engine.LoadingScene.Set(Engine.LoadingScene.argsPreLoad);
273. Engine.LoadingScene.Init();
274. CachedScenes[Name] = Engine.LoadingScene;
275. }
276. catch
277. { }
278. }
279. internal static void AddScene(Scene Source)
280. {
281. try
282. {
283. CachedScenes.Add(Source.Name, Source);
284. }
285. catch { }
286. }
287. }
288. }

#### Core/GameObject.cs

1. using DKEngine.Core.Components;
2. using System;
3. using System.Collections.Generic;
4. using System.Diagnostics;
5. using System.Drawing;
6. using System.Reflection;
7. namespace DKEngine.Core
8. {
9. /// <summary>
10. /// Primitive type for all renderable objects
11. /// </summary>
12. /// <seealso cref="DKEngine.Core.Components.Component" />
13. public class GameObject : Component
14. {
15. /// <summary>
16. /// The GameObject has shadow
17. /// </summary>
18. public bool HasShadow = false;
19. /// <summary>
20. /// Gets a value indicating whether this instance is in view.
21. /// </summary>
22. /// <value>
23. /// <c>true</c> if this instance is in view; otherwise, <c>false</c>.
24. /// </value>
25. public bool IsInView
26. {
27. get
28. {
29. float X = this.IsGUI ? 0 : Engine.BaseCam != null ? Engine.BaseCam.X : 0;
30. float Y = this.IsGUI ? 0 : Engine.BaseCam != null ? Engine.BaseCam.Y : 0;
31. return (this.Transform.Position.X + this.Transform.\_ScaledDimensions.X >= X && this.Transform.Position.X < X + Engine.Render.RenderWidth && this.Transform.Position.Y + this.Transform.\_ScaledDimensions.Y >= Y && this.Transform.Position.Y < Y + Engine.Render.RenderHeight);
32. }
33. }
34. /// <summary>
35. /// Gets or sets a value indicating whether this instance is GUI.
36. /// </summary>
37. /// <value>
38. /// <c>true</c> if this instance is GUI; otherwise, <c>false</c>.
39. /// </value>
40. public bool IsGUI
41. {
42. get { return Parent != null ? Parent.IsGUI : \_IsGUI; }
43. set { \_IsGUI = value; }
44. }
45. /// <summary>
46. /// Gets or sets the name of the type.
47. /// </summary>
48. /// <value>
49. /// The name of the type.
50. /// </value>
51. public string TypeName
52. {
53. get { return \_typeName; }
54. set
55. {
56. \_typeName = value;
57. this.Model = Database.GetGameObjectMaterial(value);
58. }
59. }
60. /// <summary>
61. /// Gets or sets the model.
62. /// </summary>
63. /// <value>
64. /// The model.
65. /// </value>
66. public Material Model
67. {
68. get { return \_Model; }
69. set
70. {
71. if (value != \_Model && value != null)
72. {
73. \_Model = value;
74. this.Transform.Dimensions = new Vector3(value.Width, value.Height, 1);
75. if (Animator?.Animations.Count == 0)
76. {
77. Animator.AddAnimation("default", \_Model);
78. Animator.Play("default");
79. }
80. }
81. }
82. }
83. /// <summary>
84. /// Gets or sets the collider.
85. /// </summary>
86. /// <value>
87. /// The collider.
88. /// </value>
89. public Collider Collider
90. {
91. get { return \_collider; }
92. set
93. {
94. if (\_collider != value)
95. {
96. if (\_collider != null)
97. {
98. foreach (Script scr in this.Scripts)
99. {
100. \_collider.CollisionEvent -= scr.CollisionHandler;
101. scr.CollisionHandler = null;
102. }
103. }
104. if (value != null)
105. {
106. foreach (Script scr in this.Scripts)
107. {
108. scr.CollisionHandler = new Collider.CollisionEnterHandler(scr.OnColliderEnter);
109. value.CollisionEvent += scr.CollisionHandler;
110. }
111. }
112. \_collider = value;
113. }
114. }
115. }
116. /// <summary>
117. /// Gets or sets the animator.
118. /// </summary>
119. /// <value>
120. /// The animator.
121. /// </value>
122. public Animator Animator { get; set; }
123. /// <summary>
124. /// Gets or sets the sound source.
125. /// </summary>
126. /// <value>
127. /// The sound source.
128. /// </value>
129. public SoundSource SoundSource { get; set; }
130. /// <summary>
131. /// Gets or sets the foreground.
132. /// </summary>
133. /// <value>
134. /// The foreground.
135. /// </value>
136. public Color? Foreground { get; set; }
137. /// <summary>
138. /// Gets the transform.
139. /// </summary>
140. /// <value>
141. /// The transform.
142. /// </value>
143. public Transform Transform { get; }
144. /// <summary>
145. /// Gets the list of childs.
146. /// </summary>
147. /// <value>
148. /// The child.
149. /// </value>
150. public List<GameObject> Child { get; }
151. internal List<Script> Scripts { get; }
152. internal bool \_IsGUI = false;
153. internal string \_typeName = "";
154. internal Material \_Model = null;
155. internal Collider \_collider = null;
156. public GameObject()
157. : base(null)
158. {
159. this.Child = new List<GameObject>();
160. this.Scripts = new List<Script>();
161. this.Transform = new Transform(this)
162. {
163. Dimensions = new Vector3(1, 1, 1),
164. Scale = new Vector3(1, 1, 1),
165. Position = new Vector3(0, 0, 0)
166. };
167. }
168. public GameObject(GameObject Parent)
169. : base(Parent)
170. {
171. this.Child = new List<GameObject>();
172. this.Scripts = new List<Script>();
173. this.Transform = new Transform(this)
174. {
175. Dimensions = new Vector3(1, 1, 1),
176. Scale = new Vector3(1, 1, 1),
177. Position = new Vector3(0, 0, 0)
178. };
179. if (Parent != null)
180. {
181. this.Parent = Parent;
182. Parent.Child.Add(this);
183. this.Transform.Position = Parent.Transform.Position;
184. this.Transform.Scale = Parent.Transform.Scale;
185. }
186. }
187. internal override void Init()
188. {
189. Initialize();
190. try
191. {
192. if (Parent == null)
193. Engine.LoadingScene.Model.Add(this);
194. Engine.LoadingScene.GameObjectsToAddToRender.Push(this);
195. }
196. catch (Exception e)
197. {
198. Debug.WriteLine("Loading scene is NULL\n\n{0}", e);
199. }
200. }
201. protected virtual void Initialize()
202. { }
203. /// <summary>
204. /// Initializes the new script.
205. /// </summary>
206. /// <typeparam name="T">Scirpt</typeparam>
207. public void InitNewScript<T>() where T : Script
208. {
209. this.Scripts.Add((T)Activator.CreateInstance(typeof(T), this));
210. }
211. /// <summary>
212. /// Initializes the new component.
213. /// </summary>
214. /// <typeparam name="T">Component</typeparam>
215. public void InitNewComponent<T>() where T : Component
216. {
217. if (typeof(T) == typeof(Animator) || typeof(T).IsSubclassOf(typeof(Animator)))
218. {
219. if (this.Animator == null)
220. {
221. this.Animator = new Animator(this);
222. }
223. return;
224. }
225. if (typeof(T) == typeof(Collider) || typeof(T).IsSubclassOf(typeof(Collider)))
226. {
227. if (this.Collider == null)
228. {
229. Type t = typeof(T);
230. this.Collider = (Collider)t.Assembly.CreateInstance(t.FullName, false, BindingFlags.Instance | BindingFlags.NonPublic | BindingFlags.Public, null, new object[] { this }, null, null);
231. }
232. return;
233. }
234. if (typeof(T) == typeof(SoundSource) || typeof(T).IsSubclassOf(typeof(SoundSource)))
235. {
236. if (this.SoundSource == null)
237. {
238. Type t = typeof(T);
239. this.SoundSource = (SoundSource)t.Assembly.CreateInstance(t.FullName, false, BindingFlags.Instance | BindingFlags.NonPublic | BindingFlags.Public, null, new object[] { this }, null, null);
240. }
241. return;
242. }
243. }
244. public override void Destroy()
245. {
246. try
247. {
248. if (Engine.LoadingScene.NewlyGeneratedComponents.Contains(this))
249. {
250. Engine.LoadingScene.DestroyObjectAwaitList.Add(this);
251. return;
252. }
253. }
254. catch { }
255. try
256. {
257. Engine.LoadingScene.AllComponents.Remove(this.Name);
258. }
259. catch { }
260. try
261. {
262. Engine.RenderObjects.Remove(this);
263. }
264. catch { }
265. try
266. {
267. Engine.LoadingScene.Model.Remove(this);
268. }
269. catch { }
270. int ScriptCount = this.Scripts.Count;
271. for (int i = 0; i < ScriptCount; i++)
272. Scripts[0].Destroy();
273. int ChildCount = this.Child.Count;
274. for (int i = 0; i < ChildCount; i++)
275. Child[0].Destroy();
276. this.Animator?.Destroy();
277. this.Animator = null;
278. this.Collider?.Destroy();
279. this.Collider = null;
280. this.Parent = null;
281. }
282. internal virtual void Render()
283. { Model?.Render(this, Foreground); }
284. /// <summary>
285. /// Finds the specified GameObject of desired name.
286. /// </summary>
287. /// <typeparam name="T">Type</typeparam>
288. /// <param name="Name">Desired name</param>
289. /// <returns></returns>
290. public static new T Find<T>(string Name) where T : GameObject
291. {
292. T retValue = null;
293. try
294. {
295. retValue = (T)Engine.LoadingScene.AllComponents[Name];
296. }
297. catch (Exception ex)
298. {
299. Debug.WriteLine("Object not found\n" + ex);
300. }
301. return retValue;
302. }
303. /// <summary>
304. /// Finds the specified GameObject of desired name.
305. /// </summary>
306. /// <param name="Name">Desired name</param>
307. /// <returns></returns>
308. public static GameObject Find(string Name)
309. {
310. GameObject retValue = null;
311. try
312. {
313. retValue = (GameObject)Engine.LoadingScene.AllComponents[Name];
314. }
315. catch (Exception ex)
316. {
317. Debug.WriteLine("Object not found\n" + ex);
318. }
319. return retValue;
320. }
321. /// <summary>
322. /// Instantiates GameObject.
323. /// </summary>
324. /// <typeparam name="T">Type</typeparam>
325. /// <param name="Position">The position</param>
326. /// <param name="Dimensions">The dimensions</param>
327. /// <param name="Scale">The scale</param>
328. /// <returns></returns>
329. public static T Instantiate<T>(Vector3 Position, Vector3 Dimensions, Vector3 Scale)
330. where T : GameObject, new()
331. {
332. T retValue = new T();
333. retValue.Transform.Position = Position;
334. retValue.Transform.Dimensions = Dimensions;
335. retValue.Transform.Scale = Scale;
336. return retValue;
337. }
338. /// <summary>
339. /// Instantiates GameObject.
340. /// </summary>
341. /// <typeparam name="T">Type</typeparam>
342. /// <param name="Transform">The transform</param>
343. /// <returns></returns>
344. public static T Instantiate<T>(Transform @Transform)
345. where T : GameObject, new()
346. {
347. return Instantiate<T>(@Transform.Position, @Transform.Dimensions, @Transform.Scale);
348. }
349. }
350. }

#### Core/Scene.cs

1. using DKEngine.Core.Components;
2. using System.Collections.Generic;
3. namespace DKEngine.Core
4. {
5. /// <summary>
6. /// DKEngine library scene
7. /// </summary>
8. /// <seealso cref="DKEngine.IPage" />
9. public abstract class Scene : IPage
10. {
11. public string Name = "";
12. internal Camera BaseCamera;
13. internal readonly Dictionary<string, Component> AllComponents;
14. internal readonly Dictionary<string, int> ComponentCount;
15. //internal readonly Dictionary<string, GameObject> AllGameObjects;
16. internal readonly List<GameObject> Model;
17. internal readonly List<Behavior> AllBehaviors;
18. internal readonly List<Collider> AllGameObjectsColliders;
19. internal readonly Stack<Component> NewlyGeneratedComponents;
20. internal readonly Stack<Behavior> NewlyGeneratedBehaviors;
21. internal readonly Stack<GameObject> GameObjectsToAddToRender;
22. internal readonly Stack<GameObject> GameObjectsAddedToRender;
23. internal readonly List<GameObject> DestroyObjectAwaitList;
24. internal object[] argsPreLoad;
25. internal object[] argsPostLoad;
26. public Scene()
27. {
28. AllComponents = new Dictionary<string, Component>(0xFFFF);
29. ComponentCount = new Dictionary<string, int>(0xFFFF);
30. AllBehaviors = new List<Behavior>(0xFFFF);
31. Model = new List<GameObject>(0xFFFF);
32. AllGameObjectsColliders = new List<Collider>(0xFFFF);
33. NewlyGeneratedComponents = new Stack<Component>(0xFFFF);
34. NewlyGeneratedBehaviors = new Stack<Behavior>(0xFFFF);
35. GameObjectsToAddToRender = new Stack<GameObject>(0xFFFF);
36. GameObjectsAddedToRender = new Stack<GameObject>(0xFFFF);
37. DestroyObjectAwaitList = new List<GameObject>(0xFFFF);
38. }
39. /// <summary>
40. /// Initializes model of Scene.
41. /// </summary>
42. public abstract void Init();
43. /// <summary>
44. /// Sets the specified arguments.
45. /// </summary>
46. /// <param name="args">The arguments</param>
47. public virtual void Set(params object[] args)
48. { }
49. /// <summary>
50. /// Unloads this instance.
51. /// </summary>
52. public abstract void Unload();
53. public static T Find<T>(string name)
54. where T : Scene
55. {
56. return (T)Database.GetScene(name);
57. }
58. }
59. }

#### Core/Script.cs

1. using DKEngine.Core.Components;
2. namespace DKEngine.Core
3. {
4. /// <summary>
5. /// Script base class
6. /// </summary>
7. /// <seealso cref="DKEngine.Core.Components.Behavior" />
8. public abstract class Script : Behavior
9. {
10. internal Collider.CollisionEnterHandler CollisionHandler;
11. public Script(GameObject Parent)
12. : base(Parent)
13. {
14. if (Parent.Collider != null)
15. {
16. CollisionHandler = new Collider.CollisionEnterHandler(OnColliderEnter);
17. Parent.Collider.CollisionEvent += CollisionHandler;
18. }
19. }
20. protected internal abstract void OnColliderEnter(Collider e);
21. public override void Destroy()
22. {
23. try
24. {
25. Engine.LoadingScene.AllComponents.Remove(this.Name);
26. }
27. catch { }
28. try
29. {
30. Engine.LoadingScene.AllBehaviors.Remove(this);
31. }
32. catch { }
33. if (UpdateHandle != null)
34. Engine.UpdateEvent -= UpdateHandle;
35. if (CollisionHandler != null)
36. Parent.Collider.CollisionEvent -= CollisionHandler;
37. Parent.Scripts.Remove(this);
38. Parent = null;
39. UpdateHandle = null;
40. }
41. }
42. }

#### Data/SplashScreen.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace DKEngine
4. {
5. internal sealed class SplashScreen : GameObject
6. {
7. public SplashScreen()
8. {
9. this.TypeName = "splashScreen";
10. this.InitNewComponent<Animator>();
11. }
12. public SplashScreen(GameObject Parent)
13. : base(Parent)
14. {
15. this.TypeName = "splashScreen";
16. this.InitNewComponent<Animator>();
17. }
18. protected override void Initialize()
19. { }
20. }
21. }

#### Data/SplashScreenScene.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace DKEngine.Data
4. {
5. internal class SplashScreenScene : Scene
6. {
7. internal SplashScreen Splash;
8. public override void Init()
9. {
10. Splash = new SplashScreen();
11. Splash.Transform.Position = new Vector3(-32, 0, 0);
12. Splash.Transform.Scale = new Vector3(0.5f, 0.5f, 0);
13. Camera splashScreenCam = new Camera();
14. }
15. public override void Unload()
16. { }
17. }
18. }

### MarIO

#### Program.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using MarIO.Assets.Scenes;
4. using MarIO.Assets.Sprites;
5. using System.Globalization;
6. namespace MarIO
7. {
8. public class Program
9. {
10. public static void Main(string[] args)
11. {
12. Engine.Init();
13. Engine.Sound.SoundVolume = 0.5f;
14. Database.LoadResources(Sprites.ResourceManager.GetResourceSet(CultureInfo.CurrentCulture, true, true));
15. Database.LoadResources(Enemies.ResourceManager.GetResourceSet(CultureInfo.CurrentCulture, true, true));
16. Engine.LoadSceneToMemory<MainMenu>();
17. Engine.LoadSceneToMemory<Level\_1\_1>();
18. Engine.LoadSceneToMemory<GameOver>();
19. Engine.LoadSceneToMemory<WorldScreen>();
21. Engine.ChangeScene(nameof(MainMenu));
22. }
23. }
24. }

#### Shared.cs

1. using DKEngine.Core.Components;
2. using DKEngine.Core.UI;
3. using MarIO.Assets.Models;
4. using MarIO.Assets.Models.Miscellaneous;
5. using System;
6. using System.Collections.Generic;
7. using System.Diagnostics;
8. namespace MarIO
9. {
10. public static class Shared
11. {
12. public static class Mechanics
13. {
14. public static SoundOutput FXPlayer;
15. public static SoundSource FXSoundSource { get { return FXPlayer.SoundSource; } }
16. private static byte \_coinsCount = 0;
17. public static string GameScoreStr { get { return string.Format($"{GameScore:00000000}"); } }
18. public static short GameScore { get; set; } = 0;
19. public static byte Lives { get; set; } = 3;
20. public static byte CoinsCount
21. {
22. get { return \_coinsCount; }
23. set
24. {
25. \_coinsCount = value;
26. if (\_coinsCount > 99)
27. {
28. Lives++;
29. \_coinsCount = 0;
30. }
31. }
32. }
33. public readonly static Stopwatch TimeCounter = new Stopwatch();
34. private readonly static TimeSpan LevelTime = new TimeSpan(0, 5, 0);
35. public static TimeSpan TimeLeft
36. {
37. get { return LevelTime - TimeCounter.Elapsed; }
38. }
39. //public static
40. public static Type LastWorldType;
41. public static Mario.State MarioCurrentState
42. {
43. get;
44. set;
45. } = Mario.State.Super;
46. public const uint OverworldBackground = 0xFF30A0DD;
47. public const uint WorldChangeBackground = 0x00000000;
48. public const int GOOMBA\_POINTS = 100;
49. public const int COIN\_SCORE = 100;
50. public const int MUSHROOM\_SCORE = 200;
51. public const int FLOWER\_SCORE = 300;
52. public const int STAR\_SCORE = 500;
53. }
54. public static class AnimatedWorldReferences
55. {
56. public static List<Block> BlocksToUpdate = new List<Block>();
57. public static List<float> BlocksStartPositions = new List<float>();
58. public static List<TextBlock> FloatingTexts = new List<TextBlock>();
59. public static List<float> FloatingTextStartPosition = new List<float>();
60. public static Stack<Block> SpecialActions = new Stack<Block>();
61. public static List<Coin> FloatingCoins = new List<Coin>();
62. public static List<float> FloatingCoinsStartPosition = new List<float>();
63. }
64. public static class Assets
65. {
66. public static class Sounds
67. {
68. public const string OVERWORLD\_THEME = @".\Assets\Sounds\Overworld\_Theme.mp3";
69. public const string MARIO\_JUMP\_FX = @".\Assets\Sounds\smb\_jump-small.mp3";
70. public const string PIPE\_ENTER\_FX = @".\Assets\Sounds\smb\_pipe.mp3";
71. public const string COIN\_GET\_FX = @".\Assets\Sounds\smb\_coin.mp3";
72. public const string UP\_1\_FX = @".\Assets\Sounds\smb\_1-up.mp3";
73. public const string BREAK\_BLOCK\_FX = @".\Assets\Sounds\smb\_breakblock.mp3";
74. public const string MARIO\_DIE\_FX = @".\Assets\Sounds\smb\_mariodie.mp3";
75. public const string POWER\_UP\_FX = @".\Assets\Sounds\smb\_powerup.mp3";
76. public const string STOMP\_FX = @".\Assets\Sounds\smb\_stomp.mp3";
77. public static readonly Sound OVERWORLD\_THEME\_SOUND = new Sound(OVERWORLD\_THEME);
78. public static readonly Sound FX\_MARIO\_JUMP\_SOUND = new Sound(MARIO\_JUMP\_FX);
79. public static readonly Sound FX\_PIPE\_ENTER\_SOUND = new Sound(PIPE\_ENTER\_FX);
80. public static readonly Sound FX\_1\_UP\_SOUND = new Sound(UP\_1\_FX);
81. public static readonly Sound FX\_BREAK\_BLOCK\_SOUND = new Sound(BREAK\_BLOCK\_FX);
82. public static readonly Sound FX\_MARIO\_DIE\_SOUND = new Sound(MARIO\_DIE\_FX);
83. public static readonly Sound FX\_POWER\_UP\_SOUND = new Sound(POWER\_UP\_FX);
84. public static readonly Sound FX\_STOMP\_SOUND = new Sound(STOMP\_FX);
85. }
86. public static class Animations
87. {
88. #region Mario
89. private const string POWERUP\_LEFT = "powerup\_left";
90. private const string POWERUP\_LEFT\_MAT = "mario\_powerup\_left";
91. private const string POWERUP\_RIGHT = "powerup\_right";
92. private const string POWERUP\_RIGHT\_MAT = "mario\_powerup\_right";
93. private const string CROUCHING\_LEFT = "crouch\_left";
94. private const string CROUCHING\_LEFT\_MAT = "mario\_crouch\_left";
95. private const string CROUCHING\_RIGHT = "crouch\_right";
96. private const string CROUCHING\_RIGHT\_MAT = "mario\_crouch\_right";
97. /\*--------------- SMALL -----------------\*/
98. public const string MARIO\_IDLE\_LEFT = "idle\_left";
99. public const string MARIO\_IDLE\_LEFT\_MAT = "mario\_left";
100. public const string MARIO\_IDLE\_RIGHT = "idle\_right";
101. public const string MARIO\_IDLE\_RIGHT\_MAT = "mario\_right";
102. public const string MARIO\_MOVE\_LEFT = "move\_left";
103. public const string MARIO\_MOVE\_LEFT\_MAT = "mario\_move\_left";
104. public const string MARIO\_MOVE\_RIGHT = "move\_right";
105. public const string MARIO\_MOVE\_RIGHT\_MAT = "mario\_move\_right";
106. public const string MARIO\_JUMP\_LEFT = "jump\_left";
107. public const string MARIO\_JUMP\_LEFT\_MAT = "mario\_jump\_left";
108. public const string MARIO\_JUMP\_RIGHT = "jump\_right";
109. public const string MARIO\_JUMP\_RIGHT\_MAT = "mario\_jump\_right";
110. public const string MARIO\_DEAD = "dead";
111. public const string MARIO\_DEAD\_MAT = "mario\_dead";
112. public const string MARIO\_CROUCHING\_LEFT = CROUCHING\_LEFT;
113. public const string MARIO\_CROUCHING\_LEFT\_MAT = MARIO\_IDLE\_LEFT\_MAT;
114. public const string MARIO\_CROUCHING\_RIGHT = CROUCHING\_RIGHT;
115. public const string MARIO\_CROUCHING\_RIGHT\_MAT = MARIO\_IDLE\_RIGHT\_MAT;
116. /\*--------------- SUPER -----------------\*/
117. public const string MARIO\_SUPER\_IDLE\_LEFT = "super\_" + MARIO\_IDLE\_LEFT;
118. public const string MARIO\_SUPER\_IDLE\_LEFT\_MAT = "super\_" + MARIO\_IDLE\_LEFT\_MAT;
119. public const string MARIO\_SUPER\_IDLE\_RIGHT = "super\_" + MARIO\_IDLE\_RIGHT;
120. public const string MARIO\_SUPER\_IDLE\_RIGHT\_MAT = "super\_" + MARIO\_IDLE\_RIGHT\_MAT;
121. public const string MARIO\_SUPER\_MOVE\_LEFT = "super\_" + MARIO\_MOVE\_LEFT;
122. public const string MARIO\_SUPER\_MOVE\_LEFT\_MAT = "super\_" + MARIO\_MOVE\_LEFT\_MAT;
123. public const string MARIO\_SUPER\_MOVE\_RIGHT = "super\_" + MARIO\_MOVE\_RIGHT;
124. public const string MARIO\_SUPER\_MOVE\_RIGHT\_MAT = "super\_" + MARIO\_MOVE\_RIGHT\_MAT;
125. public const string MARIO\_SUPER\_JUMP\_LEFT = "super\_" + MARIO\_JUMP\_LEFT;
126. public const string MARIO\_SUPER\_JUMP\_LEFT\_MAT = "super\_" + MARIO\_JUMP\_LEFT\_MAT;
127. public const string MARIO\_SUPER\_JUMP\_RIGHT = "super\_" + MARIO\_JUMP\_RIGHT;
128. public const string MARIO\_SUPER\_JUMP\_RIGHT\_MAT = "super\_" + MARIO\_JUMP\_RIGHT\_MAT;
129. public const string MARIO\_SUPER\_POWERUP\_LEFT = "super\_" + POWERUP\_LEFT;
130. public const string MARIO\_SUPER\_POWERUP\_LEFT\_MAT = "super\_" + POWERUP\_LEFT\_MAT;
131. public const string MARIO\_SUPER\_POWERUP\_RIGHT = "super\_" + POWERUP\_RIGHT;
132. public const string MARIO\_SUPER\_POWERUP\_RIGHT\_MAT = "super\_" + POWERUP\_RIGHT\_MAT;
133. public const string MARIO\_SUPER\_CROUCHING\_LEFT = "super\_" + CROUCHING\_LEFT;
134. public const string MARIO\_SUPER\_CROUCHING\_LEFT\_MAT = "super\_" + CROUCHING\_LEFT\_MAT;
135. public const string MARIO\_SUPER\_CROUCHING\_RIGHT = "super\_" + CROUCHING\_RIGHT;
136. public const string MARIO\_SUPER\_CROUCHING\_RIGHT\_MAT = "super\_" + CROUCHING\_RIGHT\_MAT;
137. /\*--------------- FIRE -----------------\*/
138. public const string MARIO\_FIRE\_IDLE\_LEFT = "fire\_" + MARIO\_IDLE\_LEFT;
139. public const string MARIO\_FIRE\_IDLE\_LEFT\_MAT = "fire\_" + MARIO\_IDLE\_LEFT\_MAT;
140. public const string MARIO\_FIRE\_IDLE\_RIGHT = "fire\_" + MARIO\_IDLE\_RIGHT;
141. public const string MARIO\_FIRE\_IDLE\_RIGHT\_MAT = "fire\_" + MARIO\_IDLE\_RIGHT\_MAT;
142. public const string MARIO\_FIRE\_MOVE\_LEFT = "fire\_" + MARIO\_MOVE\_LEFT;
143. public const string MARIO\_FIRE\_MOVE\_LEFT\_MAT = "fire\_" + MARIO\_MOVE\_LEFT\_MAT;
144. public const string MARIO\_FIRE\_MOVE\_RIGHT = "fire\_" + MARIO\_MOVE\_RIGHT;
145. public const string MARIO\_FIRE\_MOVE\_RIGHT\_MAT = "fire\_" + MARIO\_MOVE\_RIGHT\_MAT;
146. public const string MARIO\_FIRE\_JUMP\_LEFT = "fire\_" + MARIO\_JUMP\_LEFT;
147. public const string MARIO\_FIRE\_JUMP\_LEFT\_MAT = "fire\_" + MARIO\_JUMP\_LEFT\_MAT;
148. public const string MARIO\_FIRE\_JUMP\_RIGHT = "fire\_" + MARIO\_JUMP\_RIGHT;
149. public const string MARIO\_FIRE\_JUMP\_RIGHT\_MAT = "fire\_" + MARIO\_JUMP\_RIGHT\_MAT;
150. public const string MARIO\_FIRE\_POWERUP\_LEFT = "fire\_" + POWERUP\_LEFT;
151. public const string MARIO\_FIRE\_POWERUP\_LEFT\_MAT = "fire\_" + POWERUP\_LEFT\_MAT;
152. public const string MARIO\_FIRE\_POWERUP\_RIGHT = "fire\_" + POWERUP\_RIGHT;
153. public const string MARIO\_FIRE\_POWERUP\_RIGHT\_MAT = "fire\_" + POWERUP\_RIGHT\_MAT;
154. public const string MARIO\_FIRE\_CROUCHING\_LEFT = "fire\_" + CROUCHING\_LEFT;
155. public const string MARIO\_FIRE\_CROUCHING\_LEFT\_MAT = "fire\_" + CROUCHING\_LEFT\_MAT;
156. public const string MARIO\_FIRE\_CROUCHING\_RIGHT = "fire\_" + CROUCHING\_RIGHT;
157. public const string MARIO\_FIRE\_CROUCHING\_RIGHT\_MAT = "fire\_" + CROUCHING\_RIGHT\_MAT;
158. /\*--------------- INVINCIBLE -----------------\*/
159. /\*public const string MARIO\_INVINCIBLE\_IDLE\_LEFT;
160. public const string MARIO\_INVINCIBLE\_IDLE\_LEFT\_MAT;
161. public const string MARIO\_INVINCIBLE\_IDLE\_RIGHT;
162. public const string MARIO\_INVINCIBLE\_IDLE\_RIGHT\_MAT;
163. public const string MARIO\_INVINCIBLE\_MOVE\_LEFT;
164. public const string MARIO\_INVINCIBLE\_MOVE\_LEFT\_MAT;
165. public const string MARIO\_INVINCIBLE\_MOVE\_RIGHT;
166. public const string MARIO\_INVINCIBLE\_MOVE\_RIGHT\_MAT;
167. public const string MARIO\_INVINCIBLE\_JUMP\_LEFT;
168. public const string MARIO\_INVINCIBLE\_JUMP\_LEFT\_MAT;
169. public const string MARIO\_INVINCIBLE\_JUMP\_RIGHT;
170. public const string MARIO\_INVINCIBLE\_JUMP\_RIGHT\_MAT;
171. public const string MARIO\_INVINCIBLE\_DEAD;
172. public const string MARIO\_INVINCIBLE\_DEAD\_MAT;\*/
173. #endregion Mario
174. }
175. }
176. }
177. }

#### SystemExt.cs

1. using DKEngine.Core.UI;
2. using MarIO.Assets.Models;
3. using MarIO.Assets.Models.Miscellaneous;
4. using System.Drawing;
5. namespace MarIO
6. {
7. public static class SystemExt
8. {
9. public static void AddAsFloatingText(this TextBlock txBlock)
10. {
11. Shared.AnimatedWorldReferences.FloatingTexts.Add(txBlock);
12. Shared.AnimatedWorldReferences.FloatingTextStartPosition.Add(txBlock.Transform.Position.Y);
13. }
14. public static void AnimateBlockCollision(this Block block)
15. {
16. if (Shared.Mechanics.MarioCurrentState == Mario.State.Small || block.HadBonus)
17. {
18. block.State = Block.CollisionState.Up;
19. Shared.AnimatedWorldReferences.BlocksToUpdate.Add(block);
20. Shared.AnimatedWorldReferences.BlocksStartPositions.Add(block.Transform.Position.Y);
21. }
22. else
23. {
24. }
25. }
26. public static void AddAsFloatingCoin(this Coin coin)
27. {
28. Shared.AnimatedWorldReferences.FloatingCoins.Add(coin);
29. Shared.AnimatedWorldReferences.FloatingCoinsStartPosition.Add(coin.Transform.Position.Y);
30. }
31. public static Color ToColor(this uint color)
32. {
33. byte a = (byte)(color >> 24);
34. byte r = (byte)(color >> 16);
35. byte g = (byte)(color >> 8);
36. byte b = (byte)(color >> 0);
37. return Color.FromArgb(a, r, g, b);
38. }
39. }
40. }

#### Assets/Models/Miscellaneous/Coin.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Models.Miscellaneous
4. {
5. public class Coin : GameObject
6. {
7. public static Sound COIN\_FX = new Sound(Shared.Assets.Sounds.COIN\_GET\_FX);
8. public Coin()
9. { }
10. public Coin(GameObject Parent)
11. : base(Parent)
12. { }
13. protected override void Initialize()
14. {
15. this.Name = "coin";
16. //this.TypeName = "coin";
17. this.InitNewComponent<Animator>();
18. this.Animator.AddAnimation("default", Database.GetGameObjectMaterial("coin"));
19. }
20. }
21. }

#### Assets/Models/Miscellaneous/Heart.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Models.Miscellaneous
4. {
5. public class Heart : GameObject
6. {
7. public Heart()
8. { }
9. public Heart(GameObject Parent)
10. : base(Parent)
11. { }
12. protected override void Initialize()
13. {
14. this.Name = "heart";
15. //this.TypeName = "coin";
16. this.InitNewComponent<Animator>();
17. this.Animator.AddAnimation("default", Database.GetGameObjectMaterial("heart"));
18. }
19. }
20. }

#### Assets/Models/Miscellaneous/PowerUp.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using DKEngine.Core.UI;
4. using MarIO.Assets.Scripts;
5. using System;
6. using System.Drawing;
7. namespace MarIO.Assets.Models.Miscellaneous
8. {
9. public class PowerUp : GameObject
10. {
11. public Mario PlayerReference;
12. public enum PowerUpType
13. {
14. Mushroom,
15. Flower,
16. Star
17. }
18. public Action OnPickedUp { get; private set; }
19. public PowerUpType Type { get; private set; }
20. protected override void Initialize()
21. {
22. this.Name = nameof(PowerUp);
23. this.InitNewComponent<Collider>();
24. this.Collider.Area = new RectangleF(0, 0, 16, 16);
25. this.Collider.Enabled = false;
26. this.InitNewScript<PowerUpScript>();
27. switch (Shared.Mechanics.MarioCurrentState)
28. {
29. case Mario.State.Small:
30. this.TypeName = "mushroom";
31. Type = PowerUpType.Mushroom;
32. OnPickedUp = () =>
33. {
34. Shared.Mechanics.GameScore += Shared.Mechanics.MUSHROOM\_SCORE;
35. TextBlock FloatingText = new TextBlock()
36. {
37. Text = string.Format("{0}", Shared.Mechanics.MUSHROOM\_SCORE),
38. TextShadow = true
39. };
40. FloatingText.Transform.Position = this.Transform.Position;
41. FloatingText.Transform.Dimensions = new Vector3(20, 6, 0);
42. FloatingText.AddAsFloatingText();
43. PlayerReference.CurrentState = Mario.State.Super;
44. OnPickedUp = null;
45. this.Destroy();
46. };
47. break;
48. case Mario.State.Super:
49. this.TypeName = "flower";
50. Type = PowerUpType.Flower;
51. this.InitNewComponent<Animator>();
52. this.Animator.AddAnimation("default", "flower");
53. this.Animator.Play("default");
54. OnPickedUp = () =>
55. {
56. Shared.Mechanics.GameScore += Shared.Mechanics.FLOWER\_SCORE;
57. TextBlock FloatingText = new TextBlock()
58. {
59. Text = string.Format("{0}", Shared.Mechanics.FLOWER\_SCORE),
60. TextShadow = true
61. };
62. FloatingText.Transform.Position = this.Transform.Position;
63. FloatingText.Transform.Dimensions = new Vector3(20, 6, 0);
64. FloatingText.AddAsFloatingText();
65. PlayerReference.CurrentState = Mario.State.Fire;
66. OnPickedUp = null;
67. this.Destroy();
68. };
69. this.Collider.IsTrigger = true;
70. break;
71. case Mario.State.Fire:
72. case Mario.State.Invincible:
73. this.TypeName = "1-UP";
74. Type = PowerUpType.Star;
75. this.InitNewComponent<Animator>();
76. this.Animator.AddAnimation("default", "star");
77. this.Animator.Play("default");
78. OnPickedUp = () =>
79. {
80. Shared.Mechanics.GameScore += Shared.Mechanics.STAR\_SCORE;
81. TextBlock FloatingText = new TextBlock()
82. {
83. Text = string.Format("{0}", Shared.Mechanics.STAR\_SCORE),
84. TextShadow = true
85. };
86. FloatingText.Transform.Position = this.Transform.Position;
87. FloatingText.Transform.Dimensions = new Vector3(20, 6, 0);
88. FloatingText.AddAsFloatingText();
89. PlayerReference.CurrentState = Mario.State.Invincible;
90. Shared.Mechanics.Lives++;
91. OnPickedUp = null;
92. this.Destroy();
93. };
94. break;
95. default:
96. throw new Exception("JAK");
97. }
98. }
99. }
100. }

#### Assets/Models/AnimatedObject.cs

1. using DKEngine.Core;
2. namespace MarIO.Assets.Models
3. {
4. public abstract class AnimatedObject : GameObject
5. {
6. public virtual bool IsDestroyed { get; set; }
7. public bool ChangeState = false;
8. public AnimatedObject()
9. : base()
10. { }
11. public AnimatedObject(GameObject Parent)
12. : base(Parent)
13. { }
14. }
15. }

#### Assets/Models/BackgroundWorker.cs

1. using DKEngine.Core;
2. using MarIO.Assets.Scripts;
3. namespace MarIO.Assets.Models
4. {
5. public class BackgroundWorker : GameObject
6. {
7. protected override void Initialize()
8. {
9. this.InitNewScript<BlockAnimatorScript>();
10. this.InitNewScript<FloatingCoinAnimatorScript>();
11. this.InitNewScript<FloatingTextAnimatorScript>();
12. this.InitNewScript<SpecialBlocksUpdateScript>();
13. this.InitNewScript<WorldChangeManagerScript>();
14. }
15. }
16. }

#### Assets/Models/Block.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models.Miscellaneous;
4. using MarIO.Assets.Scripts;
5. using System;
6. using System.Collections.Generic;
7. using static DKEngine.Core.Components.Transform;
8. namespace MarIO.Assets.Models
9. {
10. public class Block : AnimatedObject
11. {
12. public enum BlockType
13. {
14. Ground1,
15. Ground2,
16. Ground3,
17. Ground4,
18. Bridge,
19. Bush1,
20. Bush2,
21. Bush3,
22. BushSmall,
23. CastleBig,
24. CastleSmall,
25. Cloud1,
26. Cloud2,
27. Cloud3,
28. Fence,
29. Finish,
30. Flag,
31. FlagPole,
32. Mountain,
33. NoCoin,
34. Sky,
35. Water1,
36. Water2,
37. Pipe1,
38. Pipe2,
39. Pipe3,
40. Pipe4,
41. Pipe5,
42. UnderGround1,
43. UnderGround2,
44. UnderGround3,
45. UnderGround4,
46. UnderGroundBackground1,
47. UnderGroundBackground2,
48. NumberOfObjects
49. }
50. public static Dictionary<BlockType, string> BlockTypeNames = new Dictionary<BlockType, string>()
51. {
52. { BlockType.Bridge, "bridge" },
53. { BlockType.Bush1, "bush\_01" },
54. { BlockType.Bush2, "bush\_02" },
55. { BlockType.Bush3, "bush\_03" },
56. { BlockType.BushSmall, "bush\_small" },
57. { BlockType.CastleBig, "castle\_big" },
58. { BlockType.CastleSmall, "castle\_small" },
59. { BlockType.Cloud1, "cloud\_01" },
60. { BlockType.Cloud2, "cloud\_02" },
61. { BlockType.Cloud3, "cloud\_03" },
62. { BlockType.Fence, "fence" },
63. { BlockType.Flag, "finish\_flag" },
64. { BlockType.FlagPole, "flag\_pole" },
65. { BlockType.Finish, "" },
66. { BlockType.Ground1, "block\_1\_with\_coin" },
67. { BlockType.Ground2, "block\_02" },
68. { BlockType.Ground3, "block\_03" },
69. { BlockType.Ground4, "block\_04" },
70. { BlockType.Mountain, "mountain" },
71. { BlockType.NoCoin, "block\_nocoins" },
72. { BlockType.Pipe1, "pipe\_01" },
73. { BlockType.Pipe2, "pipe\_02" },
74. { BlockType.Pipe3, "pipe\_03" },
75. { BlockType.Pipe4, "pipe\_04" },
76. { BlockType.Pipe5, "pipe\_05" },
77. { BlockType.Sky, "sky" },
78. { BlockType.UnderGround1, "underground\_block\_01" },
79. { BlockType.UnderGround2, "underground\_block\_02" },
80. { BlockType.UnderGround3, "underground\_block\_03" },
81. { BlockType.UnderGround4, "underground\_block\_04" },
82. { BlockType.UnderGroundBackground1, "background\_01" },
83. { BlockType.UnderGroundBackground2, "background\_02" },
84. { BlockType.Water1, "water\_01" },
85. { BlockType.Water2, "water\_02" },
86. };
87. public enum CollisionState
88. {
89. Stay,
90. Up,
91. Down
92. }
93. public BlockType Type { get; set; }
94. public bool InitCollider { get; set; }
95. public CollisionState State { get; set; }
96. public bool SpecialActionActivate
97. {
98. get { return \_specialAction; }
99. set
100. {
101. if (value)
102. {
103. Shared.AnimatedWorldReferences.SpecialActions.Push(this);
104. }
105. \_specialAction = value;
106. }
107. }
108. public Action SpecialAction { get; set; }
109. public Direction PipeEnterDirection { get; set; }
110. public bool CoinGot { get; set; }
111. public bool PowerUp
112. {
113. get { return \_powerUp; }
114. set
115. {
116. \_powerUp = value;
117. if (value)
118. \_hadBonus = true;
119. }
120. }
121. public byte CoinCount
122. {
123. get { return \_coinCount; }
124. set
125. {
126. \_coinCount = value;
127. if (value > 0)
128. \_hadBonus = true;
129. }
130. }
131. public bool HadBonus
132. {
133. get { return \_hadBonus; }
134. }
135. private bool \_powerUp = false;
136. private byte \_coinCount = 0;
137. private bool \_hadBonus = false;
138. private bool \_specialAction = false;
139. private SoundOutput FX\_Player;
140. public Block()
141. : base()
142. { }
143. public Block(GameObject Parent)
144. : base(Parent)
145. { }
146. protected override void Initialize()
147. {
148. this.TypeName = BlockTypeNames[Type];
149. if (InitCollider)
150. this.InitNewComponent<Collider>();
151. switch (Type)
152. {
153. case BlockType.Finish:
154. {
155. this.Transform.Dimensions = new Vector3(32, 200, 0);
156. Block part1 = new Block(this)
157. {
158. Name = string.Format("{0}\_Flag", this.Name),
159. Type = BlockType.Flag
160. };
161. Block part2 = new Block(this)
162. {
163. Name = string.Format("{0}\_Pole", this.Name),
164. Type = BlockType.FlagPole
165. };
166. part2.Transform.Position -= new Vector3(16, 0, 0);
167. }
168. break;
169. case BlockType.Pipe1:
170. {
171. PipeEnterDirection = Direction.Right;
172. this.InitNewComponent<Collider>();
173. this.Collider.IsTrigger = true;
174. this.Collider.Area = new System.Drawing.RectangleF(-1, 15, 1, 1);
175. this.InitNewScript<PipePort>();
176. Blocker block = new Blocker(this)
177. {
178. Name = string.Format("{0}\_Blocker", this.Name)
179. };
180. block.InitNewComponent<Collider>();
181. block.Collider.Area = new System.Drawing.RectangleF(0, 0, this.Transform.Dimensions.X, this.Transform.Dimensions.Y);
182. }
183. break;
184. case BlockType.Pipe3:
185. {
186. PipeEnterDirection = Direction.Down;
187. this.InitNewComponent<Collider>();
188. this.Collider.IsTrigger = true;
189. this.Collider.Area = new System.Drawing.RectangleF(15, -1, 1, 1);
190. this.InitNewScript<PipePort>();
191. Blocker block = new Blocker(this)
192. {
193. Name = string.Format("{0}\_Blocker", this.Name)
194. };
195. block.InitNewComponent<Collider>();
196. block.Collider.Area = new System.Drawing.RectangleF(0, 0, this.Transform.Dimensions.X, this.Transform.Dimensions.Y);
197. }
198. break;
199. }
200. if (CoinCount > 0 || PowerUp)
201. {
202. this.InitNewComponent<Animator>();
203. this.Animator.AddAnimation("default", this.TypeName);
204. this.Animator.AddAnimation("nocoin", BlockTypeNames[BlockType.NoCoin]);
205. }
206. FX\_Player = GameObject.Find<SoundOutput>(nameof(SoundOutput));
207. }
208. public void GetContent()
209. {
210. if (PowerUp)
211. {
212. GameObject.Instantiate<PowerUp>(new Vector3(this.Transform.Position.X + 4, this.Transform.Position.Y, this.Transform.Position.Z - 1), new Vector3(), new Vector3(1, 1, 1));
213. PowerUp = false;
214. this.Animator.Play("nocoin");
215. }
216. else if (CoinCount > 0 && !CoinGot)
217. {
218. GameObject.Instantiate<Coin>(new Vector3(this.Transform.Position.X + 4, this.Transform.Position.Y, this.Transform.Position.Z - 1), new Vector3(), new Vector3(1, 1, 1)).AddAsFloatingCoin();
219. CoinCount--;
220. Shared.Mechanics.GameScore += Shared.Mechanics.COIN\_SCORE;
221. Shared.Mechanics.FXSoundSource.PlaySound(Coin.COIN\_FX);
222. CoinGot = true;
223. if (CoinCount == 0)
224. {
225. this.Animator.Play("nocoin");
226. }
227. }
228. }
229. public void DestroyAnim()
230. { }
231. }
232. }

#### Assets/Models/Blocker.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Models
4. {
5. public class Blocker : GameObject
6. {
7. public Blocker()
8. : base()
9. { }
10. public Blocker(GameObject Parent)
11. : base(Parent)
12. { }
13. protected override void Initialize()
14. {
15. this.InitNewComponent<Collider>();
16. }
17. }
18. }

#### Assets/Models/Delayer.cs

1. using DKEngine.Core;
2. using MarIO.Assets.Scripts;
3. using System;
4. namespace MarIO.Assets.Models
5. {
6. public class Delayer : GameObject
7. {
8. public TimeSpan TimeToWait;
9. public Action CalledAction;
10. public Delayer()
11. {
12. Name = nameof(Delayer);
13. }
14. protected override void Initialize()
15. {
16. this.InitNewScript<DelayScript>();
17. }
18. }
19. }

#### Assets/Models/Enemy.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Scripts;
4. using System.Collections.Generic;
5. namespace MarIO.Assets.Models
6. {
7. public abstract class Enemy : AnimatedObject
8. {
9. public enum EnemyType
10. {
11. Goomba,
12. GoombaBlue,
13. GoombaSilver,
14. KoopaTroopa,
15. KoopaParatroopa,
16. PiranhaPlant,
17. Spiny,
18. BuzzyBeatle,
19. BuzzyBeatleBlue,
20. BuzzyBeatleSilver,
21. FireBar,
22. BulletBill,
23. BillBlasterLarge,
24. BillBlasterSmall
25. }
26. protected static Dictionary<EnemyType, string> EnemyTypeNames = new Dictionary<EnemyType, string>()
27. {
28. { EnemyType.Goomba, "goomba" },
29. { EnemyType.GoombaBlue, "" },
30. { EnemyType.GoombaSilver, "" },
31. { EnemyType.KoopaTroopa, "" },
32. { EnemyType.KoopaParatroopa, "" },
33. { EnemyType.PiranhaPlant, "" },
34. { EnemyType.Spiny, "" },
35. { EnemyType.BuzzyBeatle, "" },
36. { EnemyType.BuzzyBeatleBlue, "" },
37. { EnemyType.BuzzyBeatleSilver, "" },
38. { EnemyType.FireBar, "" },
39. { EnemyType.BulletBill, "" },
40. { EnemyType.BillBlasterLarge, "" },
41. { EnemyType.BillBlasterSmall, "" }
42. };
43. public EnemyType Type { get; set; }
44. public Enemy()
45. : base()
46. { }
47. public Enemy(GameObject Parent)
48. : base(Parent)
49. { }
50. }
51. internal class Goomba : Enemy
52. {
53. protected override void Initialize()
54. {
55. this.Name = "Goomba";
56. this.Type = EnemyType.Goomba;
57. this.InitNewComponent<Collider>();
58. this.Collider.Area = new System.Drawing.RectangleF(0, 0, 16, 16);
59. this.InitNewScript<GoombaController>();
60. this.InitNewComponent<Animator>();
61. this.Animator.AddAnimation("default", Database.GetGameObjectMaterial(EnemyTypeNames[Type]));
62. this.Animator.AddAnimation("dead", Database.GetGameObjectMaterial(EnemyTypeNames[Type] + "\_dead"));
63. }
64. }
65. }

#### Assets/Models/Group.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Models
4. {
5. public class Group : GameObject
6. {
7. public bool InitCollider = false;
8. public Group()
9. : base()
10. { }
11. public Group(GameObject Parent)
12. : base(Parent)
13. { }
14. public Block.BlockType Type { get; set; }
15. public Vector3 SizeInBlocks { get; set; }
16. protected override void Initialize()
17. {
18. Material tmp = Database.GetGameObjectMaterial(Block.BlockTypeNames[Type]);
19. this.Transform.Dimensions = new Vector3(SizeInBlocks.X \* tmp.Width, SizeInBlocks.Y \* tmp.Height, 0);
20. for (int i = 0; i < SizeInBlocks.Y; i++)
21. {
22. for (int j = 0; j < SizeInBlocks.X; j++)
23. {
24. Block newBlock = new Block(this);
25. newBlock.Type = Type;
26. newBlock.Transform.Position += new Vector3(j \* tmp.Width \* this.Transform.Scale.X, i \* tmp.Height \* this.Transform.Scale.Y, this.Transform.Position.Z);
27. newBlock.Name = string.Format("{0}\_{1}\_{2}", Name, j, i);
28. }
29. }
30. if (InitCollider)
31. this.InitNewComponent<Collider>();
32. }
33. }
34. }

#### Assets/Models/GUIUpdater.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using DKEngine.Core.UI;
4. using MarIO.Assets.Models.Miscellaneous;
5. using MarIO.Assets.Scripts;
6. namespace MarIO.Assets.Models
7. {
8. public class GUIUpdater : GameObject
9. {
10. protected override void Initialize()
11. {
12. Name = "GUI";
13. this.IsGUI = true;
14. /\*------------ TIME TEXT ----------------\*/
15. #region TIME
16. TextBlock \_time = new TextBlock(this)
17. {
18. IsGUI = true,
19. TextShadow = true,
20. Text = "TIME",
21. FontSize = 2
22. };
23. \_time.Transform.Dimensions = new Vector3(100, 20, 1);
24. \_time.Transform.Position += new Vector3(16, 4, 128);
25. TextBlock Time = new TextBlock(this)
26. {
27. Name = "txt\_Time",
28. IsGUI = true,
29. TextShadow = true,
30. Text = "",
31. FontSize = 2
32. };
33. Time.Transform.Dimensions = new Vector3(100, 20, 1);
34. Time.Transform.Position += new Vector3(22, 16, 128);
35. #endregion TIME
36. /\*------------ SCORE TEXT ----------------\*/
37. #region SCORE
38. TextBlock Score = new TextBlock(this)
39. {
40. Name = "txt\_Score",
41. Text = "",
42. IsGUI = true,
43. TextShadow = true,
44. FontSize = 2,
45. HAlignment = Text.HorizontalAlignment.Right,
46. TextHAlignment = Text.HorizontalAlignment.Right
47. };
48. Score.Transform.Dimensions = new Vector3(100, 20, 1);
49. Score.Transform.Position += new Vector3(-16, 4, 128);
50. #endregion SCORE
51. /\*------------ COINS TEXT ----------------\*/
52. #region COINS
53. Coin UICoin = new Coin(this)
54. {
55. HasShadow = true
56. };
57. UICoin.Transform.Position += new Vector3(75, 4, 128);
58. TextBlock \_coins = new TextBlock(this)
59. {
60. Name = "txt\_Coins",
61. Text = "",
62. IsGUI = true,
63. TextShadow = true,
64. FontSize = 1.5f
65. };
66. \_coins.Transform.Dimensions = new Vector3(100, 20, 1);
67. \_coins.Transform.Position += new Vector3(85, 4, 128);
68. #endregion COINS
69. /\*------------ LIVES TEXT ----------------\*/
70. #region LIVES
71. Heart UIHeart = new Heart(this)
72. {
73. HasShadow = true
74. };
75. UIHeart.Transform.Position += new Vector3(73, 16, 128);
76. TextBlock \_lives = new TextBlock(this)
77. {
78. Name = "txt\_Lives",
79. Text = "",
80. IsGUI = true,
81. TextShadow = true,
82. FontSize = 1.5f
83. };
84. \_lives.Transform.Dimensions = new Vector3(100, 20, 1);
85. \_lives.Transform.Position += new Vector3(85, 18, 128);
86. #endregion LIVES
87. /\*------------ WORLD TEXT ----------------\*/
88. #region WORLD
89. TextBlock \_world = new TextBlock(this)
90. {
91. Text = "WORLD",
92. IsGUI = true,
93. TextShadow = true,
94. FontSize = 2,
95. HAlignment = Text.HorizontalAlignment.Right,
96. TextHAlignment = Text.HorizontalAlignment.Center
97. };
98. \_world.Transform.Dimensions = new Vector3(50, 20, 1);
99. \_world.Transform.Position += new Vector3(-90, 4, 128);
100. TextBlock World = new TextBlock(this)
101. {
102. Name = "txt\_World",
103. Text = "",
104. IsGUI = true,
105. TextShadow = true,
106. FontSize = 2,
107. HAlignment = Text.HorizontalAlignment.Right,
108. TextHAlignment = Text.HorizontalAlignment.Center
109. };
110. World.Transform.Dimensions = new Vector3(50, 20, 1);
111. World.Transform.Position += new Vector3(-90, 16, 128);
112. #endregion WORLD
113. this.InitNewScript<GUIUpdateScript>();
114. }
115. }
116. }

#### Assets/Models/Mario.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Scripts;
4. using System.Drawing;
5. using static DKEngine.Core.Components.Transform;
6. using static MarIO.Shared.Assets.Animations;
7. namespace MarIO.Assets.Models
8. {
9. public class Mario : AnimatedObject
10. {
11. private State \_currentState;
12. private bool \_isDestroyed;
13. public override bool IsDestroyed
14. {
15. get { return \_isDestroyed; }
16. set
17. {
18. \_isDestroyed = value;
19. if (value)
20. CurrentState = State.Dead;
21. }
22. }
23. public bool KilledEnemy = false;
24. public Trigger LeftTrigger { get; private set; }
25. public Trigger RightTrigger { get; private set; }
26. public Trigger TopTrigger { get; private set; }
27. public Trigger BottomTrigger { get; private set; }
28. public bool InitCharacterController { get; set; }
29. public bool InitCameraController { get; set; }
30. public bool InitCollider { get; set; }
31. public State CurrentState
32. {
33. get { return \_currentState; }
34. set
35. {
36. \_currentState = value;
37. Shared.Mechanics.MarioCurrentState = value;
38. Vector3 tmp = this.Transform.Position;
39. switch (value)
40. {
41. case State.Dead:
42. case State.Small:
43. this.Collider.Area = new RectangleF(2, 0, 12, 16);
44. TopTrigger.Transform.Position = tmp.Add(2.5f, -1, 0);//new Vector3(tmp.X + 2.5f, tmp.Y - 1, tmp.Z);
45. TopTrigger.Transform.Dimensions = new Vector3(11, 1, 0);
46. RightTrigger.Transform.Position = tmp.Add(14, 0, 0); //new Vector3(tmp.X + 14, tmp.Y, tmp.Z);
47. RightTrigger.Transform.Dimensions = new Vector3(1, 14, 0);
48. LeftTrigger.Transform.Position = tmp.Add(1, 0, 0); //new Vector3(tmp.X + 1, tmp.Y, tmp.Z);
49. LeftTrigger.Transform.Dimensions = new Vector3(1, 14, 0);
50. BottomTrigger.Transform.Position = tmp.Add(1, 16, 0); //new Vector3(tmp.X + 1, tmp.Y + 16, tmp.Z);
51. BottomTrigger.Transform.Dimensions = new Vector3(14, 2, 0);
52. TopTrigger.Collider.Area = new RectangleF(0, 0, 11, 1);
53. RightTrigger.Collider.Area = new RectangleF(0, 0, 1, 14);
54. LeftTrigger.Collider.Area = new RectangleF(0, 0, 1, 14);
55. BottomTrigger.Collider.Area = new RectangleF(0, 0, 14, 2);
56. break;
57. case State.Super:
58. case State.Fire:
59. case State.Invincible:
60. this.Collider.Area = new RectangleF(0, 0, 16, 32);
61. TopTrigger.Transform.Position = tmp.Add(0.5f, -1, 0); //new Vector3(tmp.X + 0.5f, tmp.Y - 1, tmp.Z + 0);
62. TopTrigger.Transform.Dimensions = new Vector3(15, 1, 0);
63. RightTrigger.Transform.Position = tmp.Add(16, 0, 0); //new Vector3(tmp.X + 16, tmp.Y + 0, tmp.Z + 0);
64. RightTrigger.Transform.Dimensions = new Vector3(1, 30, 0);
65. LeftTrigger.Transform.Position = tmp.Add(-1, 0, 0); //new Vector3(tmp.X - 1, tmp.Y + 0, tmp.Z + 0);
66. LeftTrigger.Transform.Dimensions = new Vector3(1, 30, 0);
67. BottomTrigger.Transform.Position = tmp.Add(0, 32, 0); //new Vector3(tmp.X + 0, tmp.Y + 32, tmp.Z + 0);
68. BottomTrigger.Transform.Dimensions = new Vector3(16, 2, 0);
69. TopTrigger.Collider.Area = new RectangleF(0, 0, 15, 1);
70. RightTrigger.Collider.Area = new RectangleF(0, 0, 1, 30);
71. LeftTrigger.Collider.Area = new RectangleF(0, 0, 1, 30);
72. BottomTrigger.Collider.Area = new RectangleF(0, 0, 16, 2);
73. break;
74. default:
75. break;
76. }
77. #if DEBUG
78. TopTrigger.Model = new Material(Color.Black, TopTrigger);
79. RightTrigger.Model = new Material(Color.Black, RightTrigger);
80. LeftTrigger.Model = new Material(Color.Black, LeftTrigger);
81. BottomTrigger.Model = new Material(Color.Black, BottomTrigger);
82. #endif
83. }
84. }
85. public Movement CurrentMovement { get; set; }
86. public Direction PipeEnteredInDirection { get { return EnteredPipe.PipeEnterDirection; } }
87. public Block EnteredPipe { get; set; }
88. public WorldChangeManagerScript WorldManager { get; set; }
89. public Mario()
90. {
91. InitTriggers();
92. }
93. public Mario(GameObject Parent)
94. : base(Parent)
95. {
96. InitTriggers();
97. }
98. public enum State
99. {
100. Dead,
101. Small,
102. Super,
103. Fire,
104. Invincible
105. }
106. public enum Movement
107. {
108. Standing,
109. Crouching
110. }
111. protected override void Initialize()
112. {
113. this.Name = "Player";
114. this.InitNewComponent<Animator>();
115. this.Animator.AddAnimation(MARIO\_IDLE\_LEFT, MARIO\_IDLE\_LEFT\_MAT);
116. this.Animator.AddAnimation(MARIO\_IDLE\_RIGHT, MARIO\_IDLE\_RIGHT\_MAT);
117. this.Animator.AddAnimation(MARIO\_JUMP\_LEFT, MARIO\_JUMP\_LEFT\_MAT);
118. this.Animator.AddAnimation(MARIO\_JUMP\_RIGHT, MARIO\_JUMP\_RIGHT\_MAT);
119. this.Animator.AddAnimation(MARIO\_MOVE\_LEFT, MARIO\_MOVE\_LEFT\_MAT);
120. this.Animator.AddAnimation(MARIO\_MOVE\_RIGHT, MARIO\_MOVE\_RIGHT\_MAT);
121. this.Animator.AddAnimation(MARIO\_DEAD, MARIO\_DEAD\_MAT);
122. this.Animator.AddAnimation(MARIO\_CROUCHING\_LEFT, MARIO\_CROUCHING\_LEFT\_MAT);
123. this.Animator.AddAnimation(MARIO\_CROUCHING\_RIGHT, MARIO\_CROUCHING\_RIGHT\_MAT);
124. this.Animator.AddAnimation(MARIO\_SUPER\_IDLE\_LEFT, MARIO\_SUPER\_IDLE\_LEFT\_MAT);
125. this.Animator.AddAnimation(MARIO\_SUPER\_IDLE\_RIGHT, MARIO\_SUPER\_IDLE\_RIGHT\_MAT);
126. this.Animator.AddAnimation(MARIO\_SUPER\_JUMP\_LEFT, MARIO\_SUPER\_JUMP\_LEFT\_MAT);
127. this.Animator.AddAnimation(MARIO\_SUPER\_JUMP\_RIGHT, MARIO\_SUPER\_JUMP\_RIGHT\_MAT);
128. this.Animator.AddAnimation(MARIO\_SUPER\_MOVE\_LEFT, MARIO\_SUPER\_MOVE\_LEFT\_MAT);
129. this.Animator.AddAnimation(MARIO\_SUPER\_MOVE\_RIGHT, MARIO\_SUPER\_MOVE\_RIGHT\_MAT);
130. this.Animator.AddAnimation(MARIO\_SUPER\_POWERUP\_LEFT, MARIO\_SUPER\_POWERUP\_LEFT\_MAT);
131. this.Animator.AddAnimation(MARIO\_SUPER\_POWERUP\_RIGHT, MARIO\_SUPER\_POWERUP\_RIGHT\_MAT);
132. this.Animator.AddAnimation(MARIO\_SUPER\_CROUCHING\_RIGHT, MARIO\_SUPER\_CROUCHING\_RIGHT\_MAT);
133. this.Animator.AddAnimation(MARIO\_SUPER\_CROUCHING\_LEFT, MARIO\_SUPER\_CROUCHING\_LEFT\_MAT);
134. /\*this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_LEFT, Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_LEFT\_MAT);
135. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_RIGHT, Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_RIGHT\_MAT);
136. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_LEFT, Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_LEFT\_MAT);
137. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_RIGHT, Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_RIGHT\_MAT);
138. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_LEFT, Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_LEFT\_MAT);
139. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_RIGHT, Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_RIGHT\_MAT);
140. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_POWERUP\_LEFT, Shared.Assets.Animations.MARIO\_FIRE\_POWERUP\_LEFT\_MAT);
141. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_POWERUP\_RIGHT, Shared.Assets.Animations.MARIO\_FIRE\_POWERUP\_RIGHT\_MAT);
142. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_CROUCHING\_RIGHT, Shared.Assets.Animations.MARIO\_FIRE\_CROUCHING\_RIGHT\_MAT);
143. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_FIRE\_CROUCHING\_LEFT, Shared.Assets.Animations.MARIO\_FIRE\_CROUCHING\_LEFT\_MAT);\*/
144. /\*this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_IDLE\_LEFT, Shared.Assets.Animations.MARIO\_IDLE\_LEFT\_MAT);
145. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_IDLE\_RIGHT, Shared.Assets.Animations.MARIO\_IDLE\_RIGHT\_MAT);
146. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_JUMP\_LEFT, Shared.Assets.Animations.MARIO\_JUMP\_LEFT\_MAT);
147. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_JUMP\_RIGHT, Shared.Assets.Animations.MARIO\_JUMP\_RIGHT\_MAT);
148. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_MOVE\_LEFT, Shared.Assets.Animations.MARIO\_MOVE\_LEFT\_MAT);
149. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_MOVE\_RIGHT, Shared.Assets.Animations.MARIO\_MOVE\_RIGHT\_MAT);
150. this.Animator.AddAnimation(Shared.Assets.Animations.MARIO\_DEAD, Shared.Assets.Animations.MARIO\_DEAD\_MAT);\*/
151. if (InitCharacterController)
152. this.InitNewScript<CharacterController>();
153. if (InitCameraController)
154. this.InitNewScript<CameraController>();
155. if (InitCollider)
156. {
157. this.InitNewComponent<Collider>();
158. }
159. BottomTrigger.InitNewScript<BottomMarioChecker>();
160. LeftTrigger.InitNewScript<LeftMarioChecker>();
161. RightTrigger.InitNewScript<RightMarioChecker>();
162. TopTrigger.InitNewScript<TopMarioChecker>();
163. CurrentState = Shared.Mechanics.MarioCurrentState;
164. WorldManager = Behavior.Find<WorldChangeManagerScript>("worldManager");
165. }
166. private void InitTriggers()
167. {
168. BottomTrigger = new Trigger(this)
169. {
170. Name = "Bottom\_Trigger"
171. };
172. LeftTrigger = new Trigger(this)
173. {
174. Name = "Left\_Trigger"
175. };
176. TopTrigger = new Trigger(this)
177. {
178. Name = "Top\_Trigger"
179. };
180. RightTrigger = new Trigger(this)
181. {
182. Name = "Right\_Trigger"
183. };
184. }
185. public void PipeEnter(Block Pipe)
186. {
187. ChangeState = true;
188. EnteredPipe = Pipe;
189. }
190. }
191. }

#### Assets/Models/MusicPlayer.cs

1. using DKEngine.Core;
2. using MarIO.Assets.Scripts;
3. namespace MarIO.Assets.Models
4. {
5. public class MusicPlayer : GameObject
6. {
7. protected override void Initialize()
8. {
9. this.Name = "MusicPlayer";
10. this.InitNewScript<MusicScript>();
11. }
12. }
13. }

#### Assets/Models/SoundOutput.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Models
4. {
5. public class SoundOutput : GameObject
6. {
7. protected override void Initialize()
8. {
9. this.Name = nameof(SoundOutput);
10. this.InitNewComponent<SoundSource>();
11. Shared.Mechanics.FXPlayer = this;
12. }
13. }
14. }

#### Assets/Models/Trigger.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Models
4. {
5. public class Trigger : GameObject
6. {
7. public Trigger()
8. : base()
9. { }
10. public Trigger(GameObject Parent)
11. : base(Parent)
12. { }
13. protected override void Initialize()
14. {
15. this.InitNewComponent<Collider>();
16. this.Collider.IsTrigger = true;
17. }
18. }
19. }

#### Assets/Scenes/About.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using DKEngine.Core.UI;
5. using MarIO.Assets.Models;
6. using System;
7. using System.Collections.Generic;
8. using System.Drawing;
9. using System.Linq;
10. using System.Text;
11. using System.Threading.Tasks;
12. namespace MarIO.Assets.Scenes
13. {
14. class About : Scene
15. {
16. public override void Init()
17. {
18. new Camera()
19. {
20. BackGround = Shared.Mechanics.OverworldBackground.ToColor()
21. };
22. new Group()
23. {
24. SizeInBlocks = new Vector3(1, 20, 0),
25. Type = Block.BlockType.Ground2,
26. InitCollider = true
27. }.Transform.Position = new Vector3(0, 0, 0);
28. new Group()
29. {
30. SizeInBlocks = new Vector3(1, 20, 0),
31. Type = Block.BlockType.Ground2,
32. InitCollider = true
33. }.Transform.Position = new Vector3(48, 0, 0);
34. new Group()
35. {
36. SizeInBlocks = new Vector3(2, 1, 0),
37. Type = Block.BlockType.Ground2,
38. InitCollider = true
39. }.Transform.Position = new Vector3(16, 224, 0);
40. new Block()
41. {
42. InitCollider = true,
43. Type = Block.BlockType.Pipe3,
44. SpecialAction = GoBack
45. }.Transform.Position = new Vector3(16, 192, 1);
46. new Mario()
47. {
48. InitCollider = true,
49. InitCharacterController = true
50. }.Transform.Position = new Vector3(16, 80, 0);
51. var \_Mario = new TextBlock()
52. {
53. Foreground = Color.LawnGreen,
54. FontSize = 6,
55. HAlignment = Text.HorizontalAlignment.Center,
56. IsGUI = true,
57. Text = "MARIO",
58. TextShadow = true,
59. TextHAlignment = Text.HorizontalAlignment.Center
60. };
61. \_Mario.Transform.Position += new Vector3(30, 20, 0);
62. \_Mario.Transform.Dimensions = new Vector3(200, 30, 0);
63. var \_author = new TextBlock()
64. {
65. FontSize = 2,
66. HAlignment = Text.HorizontalAlignment.Center,
67. IsGUI = true,
68. Text = "BY DAVID KNIERADL 2017",
69. TextShadow = true,
70. TextHAlignment = Text.HorizontalAlignment.Center
71. };
72. \_author.Transform.Position += new Vector3(30, 80, 0);
73. \_author.Transform.Dimensions = new Vector3(200, 30, 0);
74. var \_using = new TextBlock()
75. {
76. Foreground = Color.YellowGreen,
77. FontSize = 3,
78. HAlignment = Text.HorizontalAlignment.Center,
79. IsGUI = true,
80. Text = "Made with",
81. TextShadow = true,
82. TextHAlignment = Text.HorizontalAlignment.Center
83. };
84. \_using.Transform.Position += new Vector3(30, 110, 0);
85. \_using.Transform.Dimensions = new Vector3(200, 30, 0);
86. var \_dkengine = new TextBlock()
87. {
88. FontSize = 2,
89. HAlignment = Text.HorizontalAlignment.Center,
90. IsGUI = true,
91. Text = "DKENGINE",
92. TextShadow = true,
93. TextHAlignment = Text.HorizontalAlignment.Center
94. };
95. \_dkengine.Transform.Position += new Vector3(30, 140, 0);
96. \_dkengine.Transform.Dimensions = new Vector3(200, 30, 0);
97. var \_naudio = new TextBlock()
98. {
99. FontSize = 2,
100. HAlignment = Text.HorizontalAlignment.Center,
101. IsGUI = true,
102. Text = "NAUDIO",
103. TextShadow = true,
104. TextHAlignment = Text.HorizontalAlignment.Center
105. };
106. \_naudio.Transform.Position += new Vector3(30, 155, 0);
107. \_naudio.Transform.Dimensions = new Vector3(200, 30, 0);
108. var \_ver = new TextBlock()
109. {
110. FontSize = 1,
111. HAlignment = Text.HorizontalAlignment.Center,
112. IsGUI = true,
113. Text = "version 0.0.1 alpha",
114. TextShadow = true,
115. TextHAlignment = Text.HorizontalAlignment.Center
116. };
117. \_ver.Transform.Position += new Vector3(30, 190, 0);
118. \_ver.Transform.Dimensions = new Vector3(200, 30, 0);
119. new MusicPlayer();
120. new SoundOutput();
121. new BackgroundWorker();
122. }
123. public override void Unload()
124. { }
125. private void GoBack()
126. {
127. Engine.ChangeScene(nameof(MainMenu), true);
128. }
129. }
130. }

#### Assets/Scenes/GameOver.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using DKEngine.Core.UI;
5. using MarIO.Assets.Models;
6. using MarIO.Assets.Models.Miscellaneous;
7. using System;
8. using System.Drawing;
9. namespace MarIO.Assets.Scenes
10. {
11. internal class GameOver : Scene
12. {
13. public GameOver()
14. {
15. Name = nameof(GameOver);
16. }
17. public override void Init()
18. {
19. TextBlock GameOver = new TextBlock()
20. {
21. FontSize = 5,
22. Foreground = Color.White,
23. HAlignment = Text.HorizontalAlignment.Center,
24. IsGUI = true,
25. Name = "tx\_GameOver",
26. Text = "GAME OVER",
27. TextHAlignment = Text.HorizontalAlignment.Center,
28. VAlignment = Text.VerticalAlignment.Center,
29. };
30. GameOver.Transform.Dimensions = new Vector3(200, 30, 0);
31. GameOver.Transform.Position += new Vector3(0, -30, 0);
32. TextBlock Score = new TextBlock()
33. {
34. FontSize = 2.5f,
35. Foreground = Color.White,
36. HAlignment = Text.HorizontalAlignment.Center,
37. IsGUI = true,
38. Name = "tx\_Score",
39. Text = Shared.Mechanics.GameScoreStr,
40. TextHAlignment = Text.HorizontalAlignment.Center,
41. VAlignment = Text.VerticalAlignment.Center
42. };
43. Score.Transform.Dimensions = new Vector3(100, 30, 0);
44. Score.Transform.Position += new Vector3(0, 30, 0);
45. GameObject holder = new GameObject();
46. holder.Transform.Position = new Vector3(136, 156, 0);
47. Coin CoinIcon = new Coin(holder)
48. {
49. IsGUI = true,
50. Name = "coin\_icon"
51. };
52. CoinIcon.Transform.Scale = new Vector3(2f, 2f, 0);
53. TextBlock Coins = new TextBlock(holder)
54. {
55. FontSize = 2.5f,
56. IsGUI = true,
57. TextHAlignment = Text.HorizontalAlignment.Center,
58. Text = string.Format($"\*{Shared.Mechanics.CoinsCount:00}")
59. };
60. Coins.Transform.Dimensions = new Vector3(40, 15, 0);
61. Coins.Transform.Position += new Vector3(12, 2, 0);
62. new Delayer()
63. {
64. CalledAction = () => Engine.LoadScene<MainMenu>(),
65. TimeToWait = new TimeSpan(0, 0, 5)
66. };
67. new Camera()
68. {
69. BackGround = Shared.Mechanics.WorldChangeBackground.ToColor()
70. };
71. Shared.Mechanics.MarioCurrentState = Mario.State.Super;
72. }
73. public override void Unload()
74. { }
75. }
76. }

#### Assets/Scenes/Level\_1\_1.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models;
4. using MarIO.Assets.Scripts;
5. namespace MarIO.Assets.Scenes
6. {
7. public class Level\_1\_1 : MapBase
8. {
9. private const int offset = 520;
10. public Level\_1\_1()
11. {
12. Name = MapBase.LevelsNames[nameof(Level\_1\_1)];
13. Shared.Mechanics.LastWorldType = typeof(Level\_1\_1);
14. }
15. public override void Load()
16. {
17. /\*-------------- BG PRESET ----------------\*/
18. for (int i = 0; i < 8; i++)
19. {
20. new Block()
21. {
22. Name = $"cloud\_1\_{i}",
23. Type = Block.BlockType.Cloud1,
24. }.Transform.Position = new Vector3(50 + i \* offset, -82, -2);
25. new Block()
26. {
27. Name = $"cloud\_2\_{i}",
28. Type = Block.BlockType.Cloud1,
29. }.Transform.Position = new Vector3(160 + i \* offset, -122, -2);
30. new Block()
31. {
32. Name = $"cloud\_3\_{i}",
33. Type = Block.BlockType.Cloud3,
34. }.Transform.Position = new Vector3(260 + i \* offset, -70, -2);
35. new Block()
36. {
37. Name = $"cloud\_4\_{i}",
38. Type = Block.BlockType.Cloud2,
39. }.Transform.Position = new Vector3(370 + i \* offset, -103, -2);
40. new Block()
41. {
42. Name = $"mountain\_1\_{i}",
43. Type = Block.BlockType.Mountain
44. }.Transform.Position = new Vector3(120 + i \* offset, 16, -2);
45. new Block()
46. {
47. Name = $"mountain\_2\_{i}",
48. Type = Block.BlockType.Mountain
49. }.Transform.Position = new Vector3(250 + i \* offset, 16, -2);
50. new Block()
51. {
52. Name = $"bush\_1\_{i}",
53. Type = Block.BlockType.Bush1
54. }.Transform.Position = new Vector3(5 + i \* offset, 24, -1);
55. new Block()
56. {
57. Name = $"bush\_2\_{i}",
58. Type = Block.BlockType.Bush3
59. }.Transform.Position = new Vector3(210 + i \* offset, 24, -1);
60. new Block()
61. {
62. Name = $"bush\_3\_{i}",
63. Type = Block.BlockType.Bush2
64. }.Transform.Position = new Vector3(375 + i \* offset, 24, -1);
65. }
66. #region GROUND
67. Group \_1 = new Group()
68. {
69. Name = "ground1",
70. InitCollider = true,
71. Type = Block.BlockType.Ground2
72. };
73. \_1.SizeInBlocks = new Vector3(64, 3, 0);
74. \_1.Transform.Position = new Vector3(0, 48, 0);
75. Group \_2 = new Group()
76. {
77. Name = "ground2",
78. InitCollider = true,
79. Type = Block.BlockType.Ground2
80. };
81. \_2.SizeInBlocks = new Vector3(20, 3, 0);
82. \_2.Transform.Position = new Vector3(1056, 48, 0);
83. Group \_3 = new Group()
84. {
85. Name = "ground3",
86. InitCollider = true,
87. Type = Block.BlockType.Ground2
88. };
89. \_3.SizeInBlocks = new Vector3(68, 3, 0);
90. \_3.Transform.Position = new Vector3(1424, 48, 0);
91. Group \_4 = new Group()
92. {
93. Name = "ground4",
94. InitCollider = true,
95. Type = Block.BlockType.Ground2
96. };
97. \_4.SizeInBlocks = new Vector3(100, 3, 0);
98. \_4.Transform.Position = new Vector3(2544, 48, 0);
99. #endregion GROUND
100. #region Platform1
101. new Block()
102. {
103. Name = "bonus\_1",
104. Type = Block.BlockType.Ground1,
105. CoinCount = 1,
106. InitCollider = true
107. }.Transform.Position = new Vector3(320, -12, 0);
108. new Block()
109. {
110. Name = "platform\_1",
111. Type = Block.BlockType.Ground4,
112. InitCollider = true
113. }.Transform.Position = new Vector3(400, -12, 0);
114. new Block()
115. {
116. Name = "platform\_1",
117. Type = Block.BlockType.Ground1,
118. InitCollider = true,
119. CoinCount = 0,
120. PowerUp = true
121. }.Transform.Position = new Vector3(416, -12, 0);
122. new Block()
123. {
124. Name = "platform\_1",
125. Type = Block.BlockType.Ground4,
126. InitCollider = true
127. }.Transform.Position = new Vector3(432, -12, 0);
128. new Block()
129. {
130. Name = "platform\_1",
131. Type = Block.BlockType.Ground1,
132. InitCollider = true,
133. CoinCount = 1
134. }.Transform.Position = new Vector3(432, -76, 0);
135. new Block()
136. {
137. Name = "platform\_1",
138. CoinCount = 1,
139. Type = Block.BlockType.Ground1,
140. InitCollider = true
141. }.Transform.Position = new Vector3(448, -12, 0);
142. new Block()
143. {
144. Name = "platform\_1",
145. Type = Block.BlockType.Ground4,
146. InitCollider = true
147. }.Transform.Position = new Vector3(464, -12, 0);
148. #endregion Platform1
149. new Block()
150. {
151. Name = "pipe",
152. Type = Block.BlockType.Pipe3
153. }.Transform.Position = new Vector3(544, 16, 1);
154. new Goomba().Transform.Position = new Vector3(600, 18, 0);
155. {
156. GameObject holder = new GameObject();
157. holder.Transform.Dimensions = new Vector3(32, 64, 0);
158. holder.Transform.Position = new Vector3(700, 32, 0);
159. holder.InitNewComponent<Collider>();
160. new Block(holder)
161. {
162. Name = "pipe",
163. Type = Block.BlockType.Pipe4
164. }.Transform.Position += new Vector3(0, 0, -1);
165. new Block(holder)
166. {
167. Name = "pipe",
168. Type = Block.BlockType.Pipe3
169. }.Transform.Position += new Vector3(0, -32, 1);
170. }
171. new Goomba().Transform.Position = new Vector3(760, 18, 0);
172. new Goomba().Transform.Position = new Vector3(800, 18, 0);
173. {
174. GameObject holder = new GameObject();
175. holder.Transform.Dimensions = new Vector3(32, 64, 0);
176. holder.Transform.Position = new Vector3(860, 16, 0);
177. holder.InitNewComponent<Collider>();
178. new Block(holder)
179. {
180. Name = "pipe",
181. Type = Block.BlockType.Pipe4
182. }.Transform.Position += new Vector3(0, 0, -1);
183. new Block(holder)
184. {
185. Name = "pipe",
186. Type = Block.BlockType.Pipe3
187. }.Transform.Position += new Vector3(0, -32, 1);
188. }
189. new Block()
190. {
191. Type = Block.BlockType.Ground4,
192. InitCollider = true
193. }.Transform.Position = new Vector3(1184, -12, 0);
194. new Block()
195. {
196. Type = Block.BlockType.Ground1,
197. CoinCount = 3,
198. InitCollider = true
199. }.Transform.Position = new Vector3(1200, -12, 0);
200. new Block()
201. {
202. Type = Block.BlockType.Ground4,
203. InitCollider = true
204. }.Transform.Position = new Vector3(1216, -12, 0);
205. for (int i = 0; i < 10; i++)
206. {
207. new Block()
208. {
209. Type = Block.BlockType.Ground4,
210. InitCollider = true
211. }.Transform.Position = new Vector3(1232 + i \* 16, -76, 0);
212. }
213. for (int i = 0; i < 3; i++)
214. {
215. new Block()
216. {
217. Type = Block.BlockType.Ground4,
218. InitCollider = true
219. }.Transform.Position = new Vector3(1440 + i \* 16, -76, 0);
220. }
221. new Block()
222. {
223. Type = Block.BlockType.Ground1,
224. InitCollider = true,
225. CoinCount = 1
226. }.Transform.Position = new Vector3(1488, -76, 0);
227. new Block()
228. {
229. Type = Block.BlockType.Ground4,
230. InitCollider = true,
231. CoinCount = 5
232. }.Transform.Position = new Vector3(1488, -12, 0);
233. new Block()
234. {
235. Type = Block.BlockType.Ground4,
236. InitCollider = true,
237. CoinCount = 5
238. }.Transform.Position = new Vector3(1616, -12, 0);
239. new Block()
240. {
241. Type = Block.BlockType.Ground4,
242. InitCollider = true,
243. CoinCount = 1
244. }.Transform.Position = new Vector3(1632, -12, 0);
245. #region Bonus Field
246. new Block()
247. {
248. Type = Block.BlockType.Ground1,
249. InitCollider = true,
250. CoinCount = 1
251. }.Transform.Position = new Vector3(1680, -12, 0);
252. new Block()
253. {
254. Type = Block.BlockType.Ground1,
255. InitCollider = true,
256. CoinCount = 1
257. }.Transform.Position = new Vector3(1744, -12, 0);
258. new Block()
259. {
260. Type = Block.BlockType.Ground1,
261. InitCollider = true,
262. PowerUp = true
263. }.Transform.Position = new Vector3(1744, -76, 0);
264. new Block()
265. {
266. Type = Block.BlockType.Ground1,
267. InitCollider = true,
268. CoinCount = 1
269. }.Transform.Position = new Vector3(1808, -12, 0);
270. #endregion Bonus Field
271. new Block()
272. {
273. Type = Block.BlockType.Ground4,
274. InitCollider = true
275. }.Transform.Position = new Vector3(1968, -12, 0);
276. for (int i = 0; i < 3; i++)
277. {
278. new Block()
279. {
280. Type = Block.BlockType.Ground4,
281. InitCollider = true
282. }.Transform.Position = new Vector3(2000 + i \* 16, -76, 0);
283. }
284. new Block()
285. {
286. Type = Block.BlockType.Ground4,
287. InitCollider = true
288. }.Transform.Position = new Vector3(2080, -76, 0);
289. new Block()
290. {
291. Type = Block.BlockType.Ground1,
292. InitCollider = true,
293. CoinCount = 1
294. }.Transform.Position = new Vector3(2096, -76, 0);
295. new Block()
296. {
297. Type = Block.BlockType.Ground1,
298. InitCollider = true,
299. CoinCount = 1
300. }.Transform.Position = new Vector3(2112, -76, 0);
301. new Block()
302. {
303. Type = Block.BlockType.Ground4,
304. InitCollider = true
305. }.Transform.Position = new Vector3(2096, -12, 0);
306. new Block()
307. {
308. Type = Block.BlockType.Ground4,
309. InitCollider = true
310. }.Transform.Position = new Vector3(2112, -12, 0);
311. new Block()
312. {
313. Type = Block.BlockType.Ground4,
314. InitCollider = true
315. }.Transform.Position = new Vector3(2128, -12, 0);
316. #region Stairs1
317. new Group()
318. {
319. InitCollider = true,
320. SizeInBlocks = new Vector3(4, 1, 0),
321. Type = Block.BlockType.Ground3
322. }.Transform.Position = new Vector3(2192, 32, 0);
323. new Group()
324. {
325. InitCollider = true,
326. SizeInBlocks = new Vector3(3, 1, 0),
327. Type = Block.BlockType.Ground3
328. }.Transform.Position = new Vector3(2208, 16, 0);
329. new Group()
330. {
331. InitCollider = true,
332. SizeInBlocks = new Vector3(2, 1, 0),
333. Type = Block.BlockType.Ground3
334. }.Transform.Position = new Vector3(2224, 0, 0);
335. new Group()
336. {
337. InitCollider = true,
338. SizeInBlocks = new Vector3(1, 1, 0),
339. Type = Block.BlockType.Ground3
340. }.Transform.Position = new Vector3(2240, -16, 0);
341. #endregion Stairs1
342. #region Stairs2
343. new Group()
344. {
345. InitCollider = true,
346. SizeInBlocks = new Vector3(4, 1, 0),
347. Type = Block.BlockType.Ground3
348. }.Transform.Position = new Vector3(2288, 32, 0);
349. new Group()
350. {
351. InitCollider = true,
352. SizeInBlocks = new Vector3(3, 1, 0),
353. Type = Block.BlockType.Ground3
354. }.Transform.Position = new Vector3(2288, 16, 0);
355. new Group()
356. {
357. InitCollider = true,
358. SizeInBlocks = new Vector3(2, 1, 0),
359. Type = Block.BlockType.Ground3
360. }.Transform.Position = new Vector3(2288, 0, 0);
361. new Group()
362. {
363. InitCollider = true,
364. SizeInBlocks = new Vector3(1, 1, 0),
365. Type = Block.BlockType.Ground3
366. }.Transform.Position = new Vector3(2288, -16, 0);
367. #endregion Stairs2
368. #region Stairs3
369. new Group()
370. {
371. InitCollider = true,
372. SizeInBlocks = new Vector3(5, 1, 0),
373. Type = Block.BlockType.Ground3
374. }.Transform.Position = new Vector3(2432, 32, 0);
375. new Group()
376. {
377. InitCollider = true,
378. SizeInBlocks = new Vector3(4, 1, 0),
379. Type = Block.BlockType.Ground3
380. }.Transform.Position = new Vector3(2448, 16, 0);
381. new Group()
382. {
383. InitCollider = true,
384. SizeInBlocks = new Vector3(3, 1, 0),
385. Type = Block.BlockType.Ground3
386. }.Transform.Position = new Vector3(2464, 0, 0);
387. new Group()
388. {
389. InitCollider = true,
390. SizeInBlocks = new Vector3(2, 1, 0),
391. Type = Block.BlockType.Ground3
392. }.Transform.Position = new Vector3(2480, -16, 0);
393. #endregion Stairs3
394. #region Stairs4
395. new Group()
396. {
397. InitCollider = true,
398. SizeInBlocks = new Vector3(4, 1, 0),
399. Type = Block.BlockType.Ground3
400. }.Transform.Position = new Vector3(2544, 32, 0);
401. new Group()
402. {
403. InitCollider = true,
404. SizeInBlocks = new Vector3(3, 1, 0),
405. Type = Block.BlockType.Ground3
406. }.Transform.Position = new Vector3(2544, 16, 0);
407. new Group()
408. {
409. InitCollider = true,
410. SizeInBlocks = new Vector3(2, 1, 0),
411. Type = Block.BlockType.Ground3
412. }.Transform.Position = new Vector3(2544, 0, 0);
413. new Group()
414. {
415. InitCollider = true,
416. SizeInBlocks = new Vector3(1, 1, 0),
417. Type = Block.BlockType.Ground3
418. }.Transform.Position = new Vector3(2544, -16, 0);
419. #endregion Stairs4
420. new Block()
421. {
422. Type = Block.BlockType.Pipe3
423. }.Transform.Position = new Vector3(2704, 16, 1);
424. new Block()
425. {
426. Type = Block.BlockType.Ground4,
427. InitCollider = true
428. }.Transform.Position = new Vector3(2768, -12, 0);
429. new Block()
430. {
431. Type = Block.BlockType.Ground4,
432. InitCollider = true
433. }.Transform.Position = new Vector3(2784, -12, 0);
434. new Block()
435. {
436. Type = Block.BlockType.Ground1,
437. InitCollider = true,
438. CoinCount = 1
439. }.Transform.Position = new Vector3(2800, -12, 0);
440. new Block()
441. {
442. Type = Block.BlockType.Ground4,
443. InitCollider = true
444. }.Transform.Position = new Vector3(2816, -12, 0);
445. new Block()
446. {
447. Type = Block.BlockType.Pipe3
448. }.Transform.Position = new Vector3(2928, 16, 1);
449. #region Stairs5
450. new Group()
451. {
452. InitCollider = true,
453. SizeInBlocks = new Vector3(7, 1, 0),
454. Type = Block.BlockType.Ground3
455. }.Transform.Position = new Vector3(2960, 32, 0);
456. new Group()
457. {
458. InitCollider = true,
459. SizeInBlocks = new Vector3(6, 1, 0),
460. Type = Block.BlockType.Ground3
461. }.Transform.Position = new Vector3(2976, 16, 0);
462. new Group()
463. {
464. InitCollider = true,
465. SizeInBlocks = new Vector3(5, 1, 0),
466. Type = Block.BlockType.Ground3
467. }.Transform.Position = new Vector3(2992, 0, 0);
468. new Group()
469. {
470. InitCollider = true,
471. SizeInBlocks = new Vector3(4, 1, 0),
472. Type = Block.BlockType.Ground3
473. }.Transform.Position = new Vector3(3008, -16, 0);
474. new Group()
475. {
476. InitCollider = true,
477. SizeInBlocks = new Vector3(3, 1, 0),
478. Type = Block.BlockType.Ground3
479. }.Transform.Position = new Vector3(3024, -32, 0);
480. new Group()
481. {
482. InitCollider = true,
483. SizeInBlocks = new Vector3(2, 1, 0),
484. Type = Block.BlockType.Ground3
485. }.Transform.Position = new Vector3(3040, -48, 0);
486. new Group()
487. {
488. InitCollider = true,
489. SizeInBlocks = new Vector3(1, 1, 0),
490. Type = Block.BlockType.Ground3
491. }.Transform.Position = new Vector3(3056, -64, 0);
492. #endregion Stairs5
493. new Block()
494. {
495. Type = Block.BlockType.CastleBig
496. }.Transform.Position = new Vector3(3216, -152, -1);
497. Trigger EndOfWorld = new Trigger();
498. EndOfWorld.Transform.Position = new Vector3(3216, -40, 0);
499. EndOfWorld.Transform.Dimensions = new Vector3(200, 80, 0);
500. EndOfWorld.InitNewScript<WorldEnd>();
501. Mario m = new Mario()
502. {
503. InitCameraController = true,
504. InitCharacterController = true,
505. InitCollider = true
506. };
507. m.Transform.Position = new Vector3(10, -10, 0);
508. Camera c = new Camera()
509. {
510. BackGround = Shared.Mechanics.OverworldBackground.ToColor()
511. };
512. new SoundOutput();
513. new GUIUpdater();
514. new BackgroundWorker();
515. new MusicPlayer();
516. Trigger DeathZone = new Trigger();
517. DeathZone.InitNewScript<DeathZoneScript>();
518. DeathZone.Transform.Dimensions = new Vector3(5000, 10, 0);
519. DeathZone.Transform.Position = new Vector3(0, 100, 0);
520. }
521. }
522. }

#### Assets/Scenes/MainMenu.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using DKEngine.Core.UI;
5. using MarIO.Assets.Models;
6. using MarIO.Assets.Scripts;
7. using System;
8. using System.Drawing;
9. namespace MarIO.Assets.Scenes
10. {
11. public class MainMenu : Scene
12. {
13. public MainMenu()
14. {
15. Name = nameof(MainMenu);
16. }
17. public override void Init()
18. {
19. Group wall4 = new Group()
20. {
21. InitCollider = true,
22. Name = "Wall\_4",
23. SizeInBlocks = new Vector3(21, 2, 0),
24. Type = Block.BlockType.Ground2
25. };
26. wall4.Transform.Position = new Vector3(0, 16 \* 13, 0);
27. Group wall5 = new Group()
28. {
29. InitCollider = true,
30. Name = "Wall\_5",
31. SizeInBlocks = new Vector3(22, 1, 0),
32. Type = Block.BlockType.Ground2
33. };
34. wall5.Transform.Position = new Vector3(0, 16 \* 9, 0);
35. Block pipe1 = new Block()
36. {
37. Name = "Pipe\_1\_Play",
38. Type = Block.BlockType.Pipe3
39. };
40. pipe1.Transform.Position = new Vector3(32, 16 \* 7, 1);
41. pipe1.SpecialAction = Play;
42. Block pipe2 = new Block()
43. {
44. Name = "Pipe\_2\_About",
45. Type = Block.BlockType.Pipe3
46. };
47. pipe2.Transform.Position = new Vector3(143, 16 \* 7, 1);
48. Block pipe3 = new Block()
49. {
50. Name = "Pipe\_3\_Exit",
51. Type = Block.BlockType.Pipe3
52. };
53. pipe3.Transform.Position = new Vector3(256, 16 \* 7, 1);
54. pipe3.SpecialAction = Exit;
55. Camera baseCam = new Camera()
56. {
57. BackGround = Shared.Mechanics.OverworldBackground.ToColor()
58. };
59. Mario player = new Mario()
60. {
61. InitCharacterController = true,
62. InitCollider = true
63. };
64. TextBlock MainMenuHeader = new TextBlock()
65. {
66. FontSize = 6,
67. HAlignment = Text.HorizontalAlignment.Center,
68. Name = "tx\_MainMenuHeader",
69. Text = "MARIO",
70. TextHAlignment = Text.HorizontalAlignment.Center,
71. TextShadow = true
72. };
73. MainMenuHeader.Transform.Position += new Vector3(0, 10, 0);
74. MainMenuHeader.Transform.Dimensions = new Vector3(200, 50, 0);
75. TextBlock PlayText = new TextBlock()
76. {
77. Name = "tx\_Play",
78. Text = "Play",
79. TextHAlignment = Text.HorizontalAlignment.Center,
80. TextShadow = true
81. };
82. PlayText.Transform.Position = new Vector3(9, 96, -1);
83. PlayText.Transform.Dimensions = new Vector3(80, 20, 0);
84. TextBlock OptionsText = new TextBlock()
85. {
86. Name = "tx\_Options",
87. Text = "About",
88. TextHAlignment = Text.HorizontalAlignment.Center,
89. TextShadow = true,
90. HAlignment = Text.HorizontalAlignment.Center
91. };
92. OptionsText.Transform.Position += new Vector3(0, 96, -1);
93. OptionsText.Transform.Dimensions = new Vector3(80, 20, 0);
94. TextBlock ExitText = new TextBlock()
95. {
96. Name = "tx\_Exit",
97. Text = "Exit",
98. TextHAlignment = Text.HorizontalAlignment.Center,
99. TextShadow = true,
100. HAlignment = Text.HorizontalAlignment.Right
101. };
102. ExitText.Transform.Position += new Vector3(-8, 96, -1);
103. ExitText.Transform.Dimensions = new Vector3(80, 20, 0);
104. Block cloud1 = new Block()
105. {
106. Name = "cloud\_1",
107. Type = Block.BlockType.Cloud3
108. };
109. cloud1.Transform.Position = new Vector3(-10, 20, -1);
110. Block cloud2 = new Block()
111. {
112. Name = "cloud\_2",
113. Type = Block.BlockType.Cloud1
114. };
115. cloud2.Transform.Position = new Vector3(120, -15, -1);
116. Block cloud3 = new Block()
117. {
118. Name = "cloud\_3",
119. Type = Block.BlockType.Cloud2
120. };
121. cloud3.Transform.Position = new Vector3(180, 34, -1);
122. Block mountain = new Block()
123. {
124. Name = "mountain",
125. Type = Block.BlockType.Mountain
126. };
127. mountain.Transform.Position = new Vector3(100, 152, -1);
128. mountain.Transform.Scale = new Vector3(2, 2, 0);
129. Block bush1 = new Block()
130. {
131. Name = "bush\_1",
132. Type = Block.BlockType.Bush3
133. };
134. bush1.Transform.Position = new Vector3(180, 182, -1);
135. Block bush2 = new Block()
136. {
137. Name = "bush\_2",
138. Type = Block.BlockType.Bush2
139. };
140. bush2.Transform.Position = new Vector3(25, 182, -1);
141. Block fence1 = new Block()
142. {
143. Name = "fence\_1",
144. Type = Block.BlockType.Fence
145. };
146. fence1.Transform.Position = new Vector3(90, 192, -1);
147. Block fence2 = new Block()
148. {
149. Name = "fence\_2",
150. Type = Block.BlockType.Fence
151. };
152. fence2.Transform.Position = new Vector3(106, 192, -1);
153. Block fence3 = new Block()
154. {
155. Name = "fence\_3",
156. Type = Block.BlockType.Fence
157. };
158. fence3.Transform.Position = new Vector3(122, 192, -1);
159. Blocker leftSide = new Blocker()
160. {
161. Name = "LeftSideBlocker"
162. };
163. leftSide.Transform.Position = new Vector3(-10, -20, 0);
164. leftSide.Transform.Dimensions = new Vector3(10, 148, 0);
165. Blocker rightSide = new Blocker()
166. {
167. Name = "LeftSideBlocker"
168. };
169. rightSide.Transform.Position = new Vector3(320, -20, 0);
170. rightSide.Transform.Dimensions = new Vector3(10, 148, 0);
171. BackgroundWorker BW = new BackgroundWorker();
172. BW.InitNewComponent<Collider>();
173. BW.Collider.Area = new RectangleF(-10, 160, 10, 30);
174. BW.Collider.IsTrigger = true;
175. BW.InitNewScript<MainMenuSpawnScript>();
176. new MusicPlayer();
177. new SoundOutput();
178. }
179. public override void Set(params object[] Args)
180. { }
181. public override void Unload()
182. { }
183. private void Exit()
184. {
185. Environment.Exit(1);
186. }
187. private void Play()
188. {
189. Shared.Mechanics.MarioCurrentState = Mario.State.Small;
190. Shared.Mechanics.CoinsCount = 0;
191. Shared.Mechanics.GameScore = 0;
192. Shared.Mechanics.Lives = 3;
193. Shared.Mechanics.TimeCounter.Reset();
194. Engine.ChangeScene(nameof(WorldScreen), true, new object[] { (Action)(() => Engine.ChangeScene(MapBase.LevelsNames[nameof(Level\_1\_1)], true)), $"world:get|{nameof(Level\_1\_1)}" });
195. }
196. }
197. }

#### Assets/Scenes/MapBase.cs

1. using DKEngine.Core;
2. using System.Collections.Generic;
3. namespace MarIO.Assets.Scenes
4. {
5. public abstract class MapBase : Scene
6. {
7. public static Dictionary<string, string> LevelsNames = new Dictionary<string, string>()
8. {
9. { nameof(Test), "test" },
10. { nameof(Level\_1\_1), "1-1" }
11. };
12. public sealed override void Init()
13. {
14. Load();
15. Shared.Mechanics.TimeCounter.Start();
16. }
17. public sealed override void Unload()
18. {
19. Shared.Mechanics.TimeCounter.Reset();
20. }
21. public abstract void Load();
22. }
23. }

#### Assets/Scenes/Test.cs

1. using DKEngine.Core.Components;
2. using MarIO.Assets.Models;
3. using MarIO.Assets.Scripts;
4. using System.Drawing;
5. namespace MarIO.Assets.Scenes
6. {
7. public class Test : MapBase
8. {
9. public static string StaticName = "test";
10. public Test()
11. {
12. Name = StaticName;
13. Shared.Mechanics.LastWorldType = typeof(Test);
14. }
15. public override void Load()
16. {
17. Group \_1 = new Group()
18. {
19. Name = "ground1",
20. InitCollider = true,
21. Type = Block.BlockType.Ground2
22. };
23. \_1.SizeInBlocks = new Vector3(50, 3, 0);
24. \_1.Transform.Position = new Vector3(0, 0, 0);
25. Group \_2 = new Group()
26. {
27. Name = "ground2",
28. InitCollider = true,
29. Type = Block.BlockType.Ground2
30. };
31. \_2.SizeInBlocks = new Vector3(10, 3, 0);
32. \_2.Transform.Position = new Vector3(60 \* 16, 0, 0);
33. Group \_3 = new Group()
34. {
35. Name = "ground3",
36. Type = Block.BlockType.Ground2,
37. InitCollider = true
38. };
39. \_3.SizeInBlocks = new Vector3(50, 3, 0);
40. \_3.Transform.Position = new Vector3(80 \* 16, 0, 0);
41. for (int i = 0; i < 10; i++)
42. {
43. Block tmp = new Block()
44. {
45. Type = Block.BlockType.Ground2,
46. Name = string.Format("PlatformTest\_{0:00}", i)
47. };
48. tmp.Transform.Position = new Vector3(80 + 16 \* i, -80, 0);
49. tmp.InitCollider = true;
50. }
51. Block pipe = new Block()
52. {
53. Name = "pipe1",
54. Type = Block.BlockType.Pipe1
55. };
56. pipe.Transform.Position = new Vector3(240, -32, 0);
57. Block blck = new Block()
58. {
59. Name = "random1",
60. Type = Block.BlockType.Ground2
61. };
62. blck.InitNewComponent<Collider>();
63. blck.Collider.Area = new System.Drawing.RectangleF(0, 0, 16, 16);
64. blck.Transform.Position = new Vector3(400, -16, 0);
65. Block blck2 = new Block()
66. {
67. Type = Block.BlockType.Ground2,
68. Name = "random2"
69. };
70. blck2.Transform.Position = new Vector3(600, -16, 0);
71. blck2.InitNewComponent<Collider>();
72. blck2.Collider.Area = new System.Drawing.RectangleF(0, 0, 16, 16);
73. Goomba goomba = new Goomba();
74. goomba.Transform.Position = new Vector3(500, -20, 0);
75. Mario m = new Mario()
76. {
77. InitCameraController = true,
78. InitCharacterController = true,
79. InitCollider = true
80. };
81. m.Transform.Position = new Vector3(10, -10, 0);
82. new MusicPlayer();
83. Camera c = new Camera()
84. {
85. BackGround = Shared.Mechanics.OverworldBackground.ToColor()
86. };
87. new GUIUpdater();
88. new SoundOutput();
89. new BackgroundWorker();
90. Trigger DeathZone = new Trigger();
91. DeathZone.InitNewScript<DeathZoneScript>();
92. DeathZone.Transform.Dimensions = new Vector3(3200, 10, 0);
93. DeathZone.Transform.Position = new Vector3(0, 50, 0);
94. DeathZone.Model = new Material(Color.Black, DeathZone);
95. }
96. }
97. }

#### Assets/Scenes/WorldScreen.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using DKEngine.Core.UI;
4. using MarIO.Assets.Models;
5. using MarIO.Assets.Models.Miscellaneous;
6. using System;
7. using System.Drawing;
8. using System.Linq;
9. namespace MarIO.Assets.Scenes
10. {
11. public class WorldScreen : Scene
12. {
13. private static readonly TimeSpan \_defautlTimeSpan = new TimeSpan(0, 0, 5);
14. private TextBlock World;
15. private TextBlock Lives;
16. private Delayer Delayer;
17. private static string RemainingLives = "";
18. private static string WorldName = "";
19. public static Action WorldChange;
20. public static TimeSpan? Delay;
21. public WorldScreen()
22. {
23. Name = nameof(WorldScreen);
24. }
25. public override void Init()
26. {
27. TextBlock \_World = new TextBlock()
28. {
29. FontSize = 5,
30. Foreground = Color.White,
31. HAlignment = Text.HorizontalAlignment.Center,
32. IsGUI = true,
33. Name = "tx\_const\_world",
34. Text = "WORLD",
35. TextHAlignment = Text.HorizontalAlignment.Center,
36. VAlignment = Text.VerticalAlignment.Center
37. };
38. \_World.Transform.Position += new Vector3(0, -40, 0);
39. \_World.Transform.Dimensions = new Vector3(120, 30, 0);
40. World = new TextBlock()
41. {
42. FontSize = 4,
43. Foreground = Color.White,
44. HAlignment = Text.HorizontalAlignment.Center,
45. IsGUI = true,
46. Name = "tx\_world",
47. TextHAlignment = Text.HorizontalAlignment.Center,
48. VAlignment = Text.VerticalAlignment.Center,
49. Text = WorldName
50. };
51. World.Transform.Position += new Vector3(0, -5, 0);
52. World.Transform.Dimensions = new Vector3(100, 30, 0);
53. GameObject holder = new GameObject();
54. holder.Transform.Position = new Vector3(120, 140, 0);
55. Heart \_HeartIcon = new Heart(holder)
56. {
57. IsGUI = true,
58. Name = "heart\_icon"
59. };
60. \_HeartIcon.Transform.Scale = new Vector3(3, 3, 0);
61. Lives = new TextBlock(holder)
62. {
63. FontSize = 3.5f,
64. IsGUI = true,
65. TextHAlignment = Text.HorizontalAlignment.Center,
66. Text = RemainingLives
67. };
68. Lives.Transform.Dimensions = new Vector3(40, 15, 0);
69. Lives.Transform.Position += new Vector3(32, 8, 0);
70. Delayer = new Delayer()
71. {
72. CalledAction = WorldChange,
73. TimeToWait = Delay ?? \_defautlTimeSpan
74. };
75. new Camera()
76. {
77. BackGround = Shared.Mechanics.WorldChangeBackground.ToColor()
78. };
79. if (Shared.Mechanics.MarioCurrentState == Mario.State.Dead)
80. Shared.Mechanics.MarioCurrentState = Mario.State.Small;
81. }
82. public override void Set(params object[] args)
83. {
84. if (args == null)
85. return;
86. string[] stringParameters = args.Where(obj => obj is string).ToList().Cast<string>().ToArray();
87. object[] otherParameters = args.Where(obj => !(obj is string)).ToArray();
88. for (int i = 0; i < stringParameters.Length; i++)
89. {
90. string[] parameters = stringParameters[i].Split(':');
91. switch (parameters[0])
92. {
93. case "world":
94. if (parameters[1].Split('|')[0] == "get")
95. {
96. WorldName = MapBase.LevelsNames[parameters[1].Split('|')[1]];
97. }
98. else
99. {
100. WorldName = parameters[1];
101. }
102. break;
103. case "time":
104. Delay = TimeSpan.Parse(parameters[1]);
105. break;
106. }
107. }
108. foreach (object item in otherParameters)
109. {
110. if (item is Action)
111. {
112. WorldChange = ((Action)item);
113. }
114. }
115. RemainingLives = string.Format($"\*{Shared.Mechanics.Lives:00}");
116. }
117. public override void Unload()
118. { }
119. }
120. }

#### Assets/Scripts/BlockAnimatorScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using MarIO.Assets.Models;
5. namespace MarIO.Assets.Scripts
6. {
7. public class BlockAnimatorScript : Script
8. {
9. private float AnimationHeight = 2;
10. private float AnimationSpeed = 20;
11. public BlockAnimatorScript(GameObject Parent)
12. : base(Parent)
13. { }
14. protected override void OnColliderEnter(Collider e)
15. {
16. }
17. protected override void Start()
18. {
19. }
20. protected override void Update()
21. {
22. if (Shared.AnimatedWorldReferences.BlocksToUpdate.Count > 0)
23. {
24. for (int i = 0; i < Shared.AnimatedWorldReferences.BlocksToUpdate.Count; i++)
25. {
26. float StartBlockY = Shared.AnimatedWorldReferences.BlocksStartPositions[i];
27. Block CurrentBlock = Shared.AnimatedWorldReferences.BlocksToUpdate[i];
28. if (CurrentBlock.State == Block.CollisionState.Up && StartBlockY - AnimationHeight < CurrentBlock.Transform.Position.Y)
29. {
30. CurrentBlock.Transform.Position -= new Vector3(0, Engine.DeltaTime \* AnimationSpeed, 0);
31. if (CurrentBlock.Transform.Position.Y <= StartBlockY - AnimationHeight)
32. {
33. CurrentBlock.State = Block.CollisionState.Down;
34. }
35. }
36. else if (CurrentBlock.State == Block.CollisionState.Down && CurrentBlock.Transform.Position.Y < StartBlockY)
37. {
38. CurrentBlock.Transform.Position += new Vector3(0, Engine.DeltaTime \* AnimationSpeed, 0);
39. if (CurrentBlock.Transform.Position.Y > StartBlockY)
40. {
41. CurrentBlock.State = Block.CollisionState.Stay;
42. CurrentBlock.Transform.Position = new Vector3(CurrentBlock.Transform.Position.X, StartBlockY, CurrentBlock.Transform.Position.Z);
43. Shared.AnimatedWorldReferences.BlocksStartPositions.RemoveAt(i);
44. Shared.AnimatedWorldReferences.BlocksToUpdate.RemoveAt(i);
45. CurrentBlock.CoinGot = false;
46. i--;
47. }
48. }
49. }
50. }
51. }
52. }
53. }

#### Assets/Scripts/CameraController.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. namespace MarIO.Assets.Scripts
5. {
6. public class CameraController : Script
7. {
8. private GameObject Player;
9. private Camera TargetCam;
10. private float PositionX;
11. private float MaxCameraDistance;
12. private Vector3 Offset;
13. public CameraController(GameObject Parent)
14. : base(Parent)
15. { }
16. protected override void OnColliderEnter(Collider e)
17. { }
18. protected override void Start()
19. {
20. MaxCameraDistance = Engine.Render.RenderWidth / 3;
21. Offset = new Vector3(20, 0, 0);
22. Player = GameObject.Find<GameObject>("Player");
23. TargetCam = Component.Find<Camera>("Camera");
24. TargetCam.Position = new Vector3(0, -160, 0);
25. }
26. protected override void Update()
27. {
28. if (Player.Transform.Position.X - TargetCam.Position.X > MaxCameraDistance)
29. {
30. TargetCam.Position += new Vector3(Player.Transform.Position.X - PositionX, 0, 0);
31. }
32. if (Player.Transform.Position.X < TargetCam.Position.X)
33. {
34. Player.Transform.Position = Player.Transform.Position.Add(TargetCam.Position.X - Player.Transform.Position.X, 0, 0);
35. }
36. PositionX = Player.Transform.Position.X;
37. }
38. }
39. }

#### Assets/Scripts/CharacterController.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using MarIO.Assets.Models;
5. using MarIO.Assets.Scenes;
6. using System;
7. using static DKEngine.Core.Components.Transform;
8. namespace MarIO.Assets.Scripts
9. {
10. public class CharacterController : Script
11. {
12. public bool Enabled = true;
13. private Animator PlayerAnimator;
14. private Mario Player;
15. //private SoundSource SoundOutput;
16. private float horiSpeed = 0;
17. private float vertSpeed = 0;
18. private const float MovementSpeed = 120f;
19. private const float FloatSpeed = 300f;
20. private const float Acceleration = 3.5f;
21. private const float DeathAnimSpeed = 120f;
22. private bool CanJump = true;
23. private bool IsFalling = false;
24. private bool Jumped = false;
25. private bool IsFacingLeft = false;
26. private bool EnemyKilledAnim = false;
27. private bool FirstTimeDeadAnimPlay = true;
28. private bool FirstTimePipeEnter = true;
29. private float PipeEnterStartPosition;
30. private float PipeEnterSpeed = 50f;
31. private readonly TimeSpan WorldReload = new TimeSpan(0, 0, 3);
32. private TimeSpan WorldReloadNow = new TimeSpan();
33. private Mario.State LastState;
34. private bool ChangingState = false;
35. private string \_idle
36. {
37. get
38. {
39. switch (Player.CurrentState)
40. {
41. case Mario.State.Dead:
42. case Mario.State.Small:
43. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_IDLE\_RIGHT;
44. case Mario.State.Super:
45. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_SUPER\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_SUPER\_IDLE\_RIGHT;
46. case Mario.State.Fire:
47. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_RIGHT;
48. /\*case Mario.State.Invincible:
49. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_IDLE\_RIGHT;\*/
50. default:
51. throw new Exception("JAK");
52. }
53. }
54. }
55. private string \_crouch
56. {
57. get
58. {
59. switch (Player.CurrentState)
60. {
61. case Mario.State.Small:
62. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_CROUCHING\_LEFT : Shared.Assets.Animations.MARIO\_CROUCHING\_RIGHT;
63. case Mario.State.Super:
64. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_SUPER\_CROUCHING\_LEFT : Shared.Assets.Animations.MARIO\_SUPER\_CROUCHING\_RIGHT;
65. case Mario.State.Fire:
66. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_FIRE\_CROUCHING\_LEFT : Shared.Assets.Animations.MARIO\_FIRE\_CROUCHING\_RIGHT;
67. /\*case Mario.State.Invincible:
68. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_IDLE\_RIGHT;\*/
69. default:
70. throw new Exception("JAK");
71. }
72. }
73. }
74. private string \_superPowerUp
75. {
76. get { return IsFacingLeft ? Shared.Assets.Animations.MARIO\_SUPER\_POWERUP\_LEFT : Shared.Assets.Animations.MARIO\_SUPER\_POWERUP\_RIGHT; }
77. }
78. private string \_firePowerUp
79. {
80. get { return IsFacingLeft ? Shared.Assets.Animations.MARIO\_FIRE\_POWERUP\_LEFT : Shared.Assets.Animations.MARIO\_FIRE\_POWERUP\_RIGHT; }
81. }
82. private string IDLE
83. {
84. get
85. {
86. return Player.CurrentMovement == Mario.Movement.Crouching ? \_crouch : \_idle;
87. }
88. }
89. private string MOVE
90. {
91. get
92. {
93. switch (Player.CurrentState)
94. {
95. case Mario.State.Small:
96. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_MOVE\_LEFT;
97. case Mario.State.Super:
98. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_SUPER\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_SUPER\_MOVE\_LEFT;
99. case Mario.State.Fire:
100. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_LEFT;
101. /\*case Mario.State.Invincible:
102. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_MOVE\_LEFT;\*/
103. default:
104. throw new Exception("JAK");
105. }
106. }
107. }
108. private string JUMP
109. {
110. get
111. {
112. switch (Player.CurrentState)
113. {
114. case Mario.State.Small:
115. return horiSpeed != 0 ? (horiSpeed > 0 ? Shared.Assets.Animations.MARIO\_JUMP\_RIGHT : Shared.Assets.Animations.MARIO\_JUMP\_LEFT)
116. : (IsFacingLeft ? Shared.Assets.Animations.MARIO\_JUMP\_LEFT : Shared.Assets.Animations.MARIO\_JUMP\_RIGHT);
117. case Mario.State.Super:
118. return horiSpeed != 0 ? (horiSpeed > 0 ? Shared.Assets.Animations.MARIO\_SUPER\_JUMP\_RIGHT : Shared.Assets.Animations.MARIO\_SUPER\_JUMP\_LEFT)
119. : (IsFacingLeft ? Shared.Assets.Animations.MARIO\_SUPER\_JUMP\_LEFT : Shared.Assets.Animations.MARIO\_SUPER\_JUMP\_RIGHT);
120. case Mario.State.Fire:
121. return horiSpeed != 0 ? (horiSpeed > 0 ? Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_RIGHT : Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_LEFT)
122. : (IsFacingLeft ? Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_LEFT : Shared.Assets.Animations.MARIO\_FIRE\_JUMP\_RIGHT);
123. /\*case Mario.State.Invincible:
124. return horiSpeed != 0 ? (horiSpeed > 0 ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_JUMP\_RIGHT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_JUMP\_LEFT)
125. : (IsFacingLeft ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_JUMP\_LEFT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_JUMP\_RIGHT);\*/
126. default:
127. throw new Exception("JAK");
128. }
129. }
130. }
131. private string POWERUP
132. {
133. get
134. {
135. switch (LastState)
136. {
137. case Mario.State.Small:
138. return \_superPowerUp;
139. case Mario.State.Super:
140. return LastState < Player.CurrentState ? \_firePowerUp : \_superPowerUp;
141. case Mario.State.Fire:
142. return LastState < Player.CurrentState ? "" : \_firePowerUp;
143. default:
144. throw new Exception("JAK");
145. }
146. }
147. }
148. public CharacterController(GameObject Parent)
149. : base(Parent)
150. {
151. this.Name = nameof(CharacterController);
152. this.Parent.InitNewComponent<Collider>();
153. }
154. protected override void OnColliderEnter(Collider e)
155. { }
156. protected override void Start()
157. {
158. Player = GameObject.Find<Mario>("Player");
159. PlayerAnimator = Component.Find<Animator>("Player\_Animator");
160. //SoundOutput = Component.Find<SoundSource>("Player\_SoundSource");
161. LastState = Player.CurrentState;
162. Player.Animator.Play(Shared.Assets.Animations.MARIO\_IDLE\_RIGHT);
163. }
164. protected override void Update()
165. {
166. if (!Enabled)
167. return;
168. if (LastState != Player.CurrentState && Player.CurrentState != Mario.State.Dead)
169. {
170. if (!ChangingState)
171. {
172. PlayerAnimator.Play(POWERUP);
173. Shared.Mechanics.FXSoundSource.PlaySound(Shared.Assets.Sounds.FX\_POWER\_UP\_SOUND);
174. bool FromSmalltoLarge = Player.CurrentState > Mario.State.Small && LastState == Mario.State.Small;
175. bool FromLargeToSmall = Player.CurrentState == Mario.State.Small && LastState == Mario.State.Super;
176. float YtoAdd = FromSmalltoLarge ? -16 : (FromLargeToSmall ? 0 : 16);
177. Player.Transform.Position += new Vector3(0, YtoAdd, 0);
178. ChangingState = true;
179. Player.LeftTrigger.Collider.Enabled = false;
180. Player.RightTrigger.Collider.Enabled = false;
181. Player.TopTrigger.Collider.Enabled = false;
182. Player.BottomTrigger.Collider.Enabled = false;
183. Player.Collider.Enabled = false;
184. return;
185. }
186. if (PlayerAnimator.NumberOfPlays > 5)
187. {
188. LastState = Player.CurrentState;
189. Player.LeftTrigger.Collider.Enabled = true;
190. Player.RightTrigger.Collider.Enabled = true;
191. Player.TopTrigger.Collider.Enabled = true;
192. Player.BottomTrigger.Collider.Enabled = true;
193. Player.Collider.Enabled = true;
194. ChangingState = false;
195. }
196. else
197. return;
198. }
199. else if (Player.CurrentState == Mario.State.Dead)
200. {
201. DeadAnimation();
202. }
203. else if (Player.KilledEnemy)
204. {
205. Shared.Mechanics.FXSoundSource.PlaySound(Shared.Assets.Sounds.FX\_STOMP\_SOUND);
206. Player.KilledEnemy = false;
207. EnemyKilledAnim = true;
208. Jumped = true;
209. IsFalling = false;
210. vertSpeed = -FloatSpeed;
211. }
212. else if (Player.ChangeState)
213. {
214. if (FirstTimePipeEnter)
215. {
216. Shared.Mechanics.FXSoundSource.StopSound(Shared.Assets.Sounds.OVERWORLD\_THEME\_SOUND);
217. Shared.Mechanics.FXSoundSource.PlaySound(Shared.Assets.Sounds.FX\_PIPE\_ENTER\_SOUND);
218. Player.Collider.Enabled = false;
219. PipeEnterStartPosition = Player.PipeEnteredInDirection == Direction.Down ? Player.Transform.Position.Y : Player.Transform.Position.X;
220. horiSpeed = 0;
221. vertSpeed = 0;
222. FirstTimePipeEnter = false;
223. }
224. if (Player.PipeEnteredInDirection == Direction.Right)
225. {
226. if (Player.Transform.Position.X < PipeEnterStartPosition + 16)
227. {
228. horiSpeed = PipeEnterSpeed;
229. }
230. else
231. {
232. Player.WorldManager.CurrentlyEnteredPipeScript = Player.EnteredPipe;
233. }
234. }
235. else if (Player.PipeEnteredInDirection == Direction.Down)
236. {
237. if (Player.Transform.Position.Y < PipeEnterStartPosition + 16)
238. {
239. vertSpeed = PipeEnterSpeed;
240. }
241. else
242. {
243. Player.WorldManager.CurrentlyEnteredPipeScript = Player.EnteredPipe;
244. }
245. }
246. }
247. else if (Player.CurrentState > Mario.State.Dead)
248. {
249. Movement();
250. }
251. Player.Transform.Position = Player.Transform.Position.Add(horiSpeed \* Engine.DeltaTime, vertSpeed \* Engine.DeltaTime, 0);
252. AnimationControl();
253. }
254. private void DeadAnimation()
255. {
256. horiSpeed = 0;
257. if (FirstTimeDeadAnimPlay)
258. {
259. Player.Collider.Enabled = false;
260. Player.BottomTrigger.Collider.Enabled = false;
261. Player.LeftTrigger.Collider.Enabled = false;
262. Player.RightTrigger.Collider.Enabled = false;
263. Player.TopTrigger.Collider.Enabled = false;
264. vertSpeed = -FloatSpeed;
265. FirstTimeDeadAnimPlay = false;
266. Shared.Mechanics.FXSoundSource.StopSound(Shared.Assets.Sounds.OVERWORLD\_THEME\_SOUND);
267. Shared.Mechanics.FXSoundSource.PlaySound(Shared.Assets.Sounds.FX\_MARIO\_DIE\_SOUND);
268. }
269. else
270. {
271. vertSpeed += Engine.DeltaTime \* DeathAnimSpeed \* Acceleration;
272. WorldReloadNow += new TimeSpan(0, 0, 0, 0, (int)(Engine.DeltaTime \* 1000));
273. if (WorldReloadNow > WorldReload)
274. {
275. Shared.Mechanics.Lives--;
276. if (Shared.Mechanics.Lives == 0)
277. Engine.ChangeScene(nameof(GameOver), true);
278. else
279. Engine.ChangeScene(nameof(WorldScreen), true);
280. }
281. }
282. }
283. private void Movement()
284. {
285. if (Player.Collider.Collision(Direction.Down))
286. {
287. IsFalling = false;
288. Jumped = false;
289. vertSpeed = 0;
290. }
291. if (Engine.Input.IsKeyDown(ConsoleKey.A) || horiSpeed < 0)
292. {
293. Left();
294. }
295. if (Engine.Input.IsKeyDown(ConsoleKey.W) || Jumped)
296. {
297. Jump();
298. }
299. if (Engine.Input.IsKeyDown(ConsoleKey.D) || horiSpeed > 0)
300. {
301. Right();
302. }
303. if (Engine.Input.IsKeyDown(ConsoleKey.S))
304. {
305. if (vertSpeed == 0)
306. {
307. horiSpeed = 0;
308. Player.CurrentMovement = Mario.Movement.Crouching;
309. }
310. }
311. else
312. {
313. Player.CurrentMovement = Mario.Movement.Standing;
314. }
315. if (!Player.Collider.Collision(Direction.Down))
316. {
317. Fall();
318. }
319. }
320. private void Jump()
321. {
322. if (Engine.Input.IsKeyDown(ConsoleKey.W))
323. {
324. Player.CurrentMovement = Mario.Movement.Standing;
325. if (CanJump)
326. {
327. if (EnemyKilledAnim)
328. {
329. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed \* 2;
330. if (vertSpeed <= 0)
331. {
332. IsFalling = true;
333. EnemyKilledAnim = false;
334. }
335. }
336. else if (!IsFalling)
337. {
338. if (vertSpeed == 0 && !Jumped)
339. {
340. Shared.Mechanics.FXSoundSource.PlaySound(Shared.Assets.Sounds.FX\_MARIO\_JUMP\_SOUND);
341. vertSpeed = -FloatSpeed \* 1.5f;
342. Jumped = true;
343. }
344. else if (!Player.Collider.Collision(Direction.Up) && vertSpeed < 0)
345. {
346. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed;
347. }
348. else
349. {
350. vertSpeed = 0;
351. IsFalling = true;
352. }
353. }
354. }
355. }
356. else if (Jumped)
357. {
358. if (EnemyKilledAnim)
359. {
360. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed \* 4;
361. if (vertSpeed <= 0)
362. {
363. IsFalling = true;
364. EnemyKilledAnim = false;
365. }
366. }
367. else if (!IsFalling)
368. {
369. vertSpeed = -vertSpeed;
370. IsFalling = true;
371. EnemyKilledAnim = false;
372. }
373. }
374. }
375. private void Left()
376. {
377. if (Engine.Input.IsKeyDown(ConsoleKey.A))
378. {
379. Player.CurrentMovement = Mario.Movement.Standing;
380. IsFacingLeft = true;
381. if (!Player.Collider.Collision(Direction.Left) && horiSpeed > -MovementSpeed)
382. {
383. horiSpeed -= Engine.DeltaTime \* Acceleration \* MovementSpeed;
384. }
385. else if (Player.Collider.Collision(Direction.Left))
386. {
387. horiSpeed = 0;
388. }
389. else
390. {
391. horiSpeed = -MovementSpeed;
392. }
393. }
394. else if (horiSpeed < 0)
395. {
396. IsFacingLeft = true;
397. horiSpeed += Engine.DeltaTime \* Acceleration \* MovementSpeed \* 4;
398. if (horiSpeed >= 0 || Player.Collider.Collision(Direction.Left))
399. {
400. horiSpeed = 0;
401. }
402. }
403. }
404. private void Right()
405. {
406. if (Engine.Input.IsKeyDown(ConsoleKey.D))
407. {
408. Player.CurrentMovement = Mario.Movement.Standing;
409. IsFacingLeft = false;
410. if (!Player.Collider.Collision(Direction.Right) && horiSpeed < MovementSpeed)
411. {
412. horiSpeed += Engine.DeltaTime \* Acceleration \* MovementSpeed;
413. }
414. else if (Player.Collider.Collision(Direction.Right))
415. {
416. horiSpeed = 0;
417. }
418. else
419. {
420. horiSpeed = MovementSpeed;
421. }
422. }
423. else if (horiSpeed > 0)
424. {
425. IsFacingLeft = false;
426. horiSpeed -= Engine.DeltaTime \* Acceleration \* MovementSpeed \* 2;
427. if (horiSpeed <= 0 || Player.Collider.Collision(Direction.Right))
428. {
429. horiSpeed = 0;
430. }
431. }
432. }
433. private void Fall()
434. {
435. if (!IsFalling && !Jumped)
436. {
437. vertSpeed = 0;
438. Jumped = true;
439. IsFalling = true;
440. }
441. else if (IsFalling)
442. {
443. if (vertSpeed < FloatSpeed)
444. {
445. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed;
446. if (vertSpeed > FloatSpeed)
447. vertSpeed = FloatSpeed;
448. }
449. else
450. {
451. vertSpeed = FloatSpeed;
452. }
453. }
454. }
455. private void AnimationControl()
456. {
457. if (Player.CurrentState > Mario.State.Dead)
458. {
459. if (Jumped)
460. {
461. PlayerAnimator.Play(JUMP);
462. }
463. else
464. {
465. if (horiSpeed != 0)
466. PlayerAnimator.Play(MOVE);
467. else
468. PlayerAnimator.Play(IDLE);
469. }
470. }
471. else
472. {
473. PlayerAnimator.Play(Shared.Assets.Animations.MARIO\_DEAD);
474. }
475. }
476. }
477. }

#### Assets/Scripts/DeathZoneScript.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models;
4. using System.Diagnostics;
5. namespace MarIO.Assets.Scripts
6. {
7. public class DeathZoneScript : Script
8. {
9. public DeathZoneScript(GameObject Parent) : base(Parent)
10. { }
11. protected override void OnColliderEnter(Collider e)
12. {
13. Debug.WriteLine($"{e.Parent}");
14. if (e.Parent is AnimatedObject)
15. {
16. ((AnimatedObject)e.Parent).IsDestroyed = true;
17. }
18. }
19. protected override void Start()
20. { }
21. protected override void Update()
22. { }
23. }
24. }

#### Assets/Scripts/DelayScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using MarIO.Assets.Models;
5. using System;
6. namespace MarIO.Assets.Scripts
7. {
8. public class DelayScript : Script
9. {
10. private TimeSpan Checker;
11. private Delayer Source;
12. public DelayScript(GameObject Parent) : base(Parent)
13. {
14. Source = (Delayer)Parent;
15. }
16. protected override void OnColliderEnter(Collider e)
17. { }
18. protected override void Start()
19. {
20. Checker = new TimeSpan();
21. }
22. protected override void Update()
23. {
24. Checker += new TimeSpan(0, 0, 0, 0, (int)(Engine.DeltaTime \* 1000));
25. if (Checker > Source?.TimeToWait)
26. {
27. Source?.CalledAction?.Invoke();
28. }
29. }
30. }
31. }

#### Assets/Scripts/EnemyControllerScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using DKEngine.Core.UI;
5. using MarIO.Assets.Models;
6. using static DKEngine.Core.Components.Transform;
7. namespace MarIO.Assets.Scripts
8. {
9. public class GoombaController : Script
10. {
11. private const int Speed = 20;
12. private const int FloatSpeed = 60;
13. private const int Acceleration = 20;
14. private int CurrentSpeed = 0;
15. private float vertSpeed = 0;
16. private bool IsFalling = false;
17. private bool firstTimeDeadAnimation = true;
18. private float DeadTimeCurrent = 0f;
19. private const float DeadTime = 3f;
20. private Enemy Target;
21. public GoombaController(GameObject Parent) : base(Parent)
22. {
23. Target = (Enemy)Parent;
24. }
25. protected override void OnColliderEnter(Collider e)
26. { }
27. protected override void Start()
28. {
29. CurrentSpeed = -Speed;
30. }
31. protected override void Update()
32. {
33. if (!Target.IsDestroyed)
34. {
35. Movement();
36. }
37. else
38. {
39. DeadAnimation();
40. }
41. }
42. private void Movement()
43. {
44. if (Target.Collider.Collision(Direction.Left))
45. {
46. CurrentSpeed = Speed;
47. }
48. if (Target.Collider.Collision(Direction.Right))
49. {
50. CurrentSpeed = -Speed;
51. }
52. if (!Target.Collider.Collision(Direction.Down))
53. {
54. if (!IsFalling)
55. {
56. vertSpeed = 0;
57. IsFalling = true;
58. }
59. else
60. {
61. if (vertSpeed < FloatSpeed)
62. {
63. vertSpeed += Engine.DeltaTime \* Acceleration;
64. }
65. else
66. {
67. vertSpeed = FloatSpeed;
68. }
69. }
70. }
71. else if (IsFalling)
72. {
73. vertSpeed = 0;
74. IsFalling = false;
75. }
76. Target.Transform.Position += new Vector3(CurrentSpeed \* Engine.DeltaTime, vertSpeed \* Engine.DeltaTime, 0);
77. }
78. private void DeadAnimation()
79. {
80. if (firstTimeDeadAnimation)
81. {
82. Shared.Mechanics.GameScore += Shared.Mechanics.GOOMBA\_POINTS;
83. TextBlock FloatingText = new TextBlock()
84. {
85. Text = string.Format("{0}", Shared.Mechanics.GOOMBA\_POINTS),
86. TextShadow = true
87. };
88. FloatingText.Transform.Position = Target.Transform.Position;
89. FloatingText.Transform.Dimensions = new Vector3(20, 6, 0);
90. FloatingText.AddAsFloatingText();
91. Target.Collider.Enabled = false;
92. Target.Animator.Play("dead");
93. firstTimeDeadAnimation = false;
94. Target.Transform.Position += new Vector3(0, 8, 0);
95. }
96. DeadTimeCurrent += Engine.DeltaTime;
97. if (DeadTimeCurrent > DeadTime)
98. {
99. Target.Destroy();
100. }
101. }
102. }
103. }

#### Assets/Scripts/FloatingCoinAnimatorScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using MarIO.Assets.Models.Miscellaneous;
5. namespace MarIO.Assets.Scripts
6. {
7. public class FloatingCoinAnimatorScript : Script
8. {
9. private float AnimationHeight = 60;
10. private float AnimationSpeed = 20;
11. public FloatingCoinAnimatorScript(GameObject Parent)
12. : base(Parent)
13. { }
14. protected override void OnColliderEnter(Collider e)
15. { }
16. protected override void Start()
17. { }
18. protected override void Update()
19. {
20. if (Shared.AnimatedWorldReferences.FloatingCoins.Count > 0)
21. {
22. for (int i = 0; i < Shared.AnimatedWorldReferences.FloatingCoins.Count; i++)
23. {
24. Coin currentCoin = Shared.AnimatedWorldReferences.FloatingCoins[i];
25. float currentCoinStartPosition = Shared.AnimatedWorldReferences.FloatingCoinsStartPosition[i];
26. if (currentCoin.Transform.Position.Y > currentCoinStartPosition - AnimationHeight)
27. {
28. currentCoin.Transform.Position -= new Vector3(0, Engine.DeltaTime \* AnimationSpeed, 0);
29. if (currentCoin.Transform.Position.Y <= currentCoinStartPosition - AnimationHeight)
30. {
31. currentCoin.Destroy();
32. Shared.AnimatedWorldReferences.FloatingCoins.RemoveAt(i);
33. Shared.AnimatedWorldReferences.FloatingCoinsStartPosition.RemoveAt(i);
34. i--;
35. }
36. }
37. }
38. }
39. }
40. }
41. }

#### Assets/Scripts/FloatingTextAnimatorScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using DKEngine.Core.UI;
5. namespace MarIO.Assets.Scripts
6. {
7. public class FloatingTextAnimatorScript : Script
8. {
9. private float AnimationHeight = 30;
10. private float AnimationSpeed = 20;
11. public FloatingTextAnimatorScript(GameObject Parent)
12. : base(Parent)
13. { }
14. protected override void OnColliderEnter(Collider e)
15. { }
16. protected override void Start()
17. { }
18. protected override void Update()
19. {
20. if (Shared.AnimatedWorldReferences.FloatingTexts.Count > 0)
21. {
22. for (int i = 0; i < Shared.AnimatedWorldReferences.FloatingTexts.Count; i++)
23. {
24. float StartTextBlockY = Shared.AnimatedWorldReferences.FloatingTextStartPosition[i];
25. TextBlock CurrentTextBlock = Shared.AnimatedWorldReferences.FloatingTexts[i];
26. if (CurrentTextBlock.Transform.Position.Y > StartTextBlockY - AnimationHeight)
27. {
28. CurrentTextBlock.Transform.Position -= new Vector3(0, Engine.DeltaTime \* AnimationSpeed, 0);
29. if (CurrentTextBlock.Transform.Position.Y < StartTextBlockY - AnimationHeight)
30. {
31. Shared.AnimatedWorldReferences.FloatingTextStartPosition.RemoveAt(i);
32. Shared.AnimatedWorldReferences.FloatingTexts.RemoveAt(i);
33. CurrentTextBlock.Destroy();
34. i--;
35. }
36. }
37. }
38. }
39. }
40. }
41. }

#### Assets/Scripts/GUIUpdateScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using DKEngine.Core.UI;
5. namespace MarIO.Assets.Scripts
6. {
7. public class GUIUpdateScript : Script
8. {
9. private TextBlock Time;
10. private TextBlock Coins;
11. private TextBlock World;
12. private TextBlock Lives;
13. private TextBlock Score;
14. public GUIUpdateScript(GameObject Parent) : base(Parent)
15. { }
16. protected override void OnColliderEnter(Collider e)
17. { }
18. protected override void Start()
19. {
20. this.World = GameObject.Find<TextBlock>("txt\_World");
21. this.Time = GameObject.Find<TextBlock>("txt\_Time");
22. this.Score = GameObject.Find<TextBlock>("txt\_Score");
23. this.Coins = GameObject.Find<TextBlock>("txt\_Coins");
24. this.Lives = GameObject.Find<TextBlock>("txt\_Lives");
25. this.World.Text = Engine.SceneName;
26. this.Time.Text = string.Format("{0:000}", Shared.Mechanics.TimeLeft.TotalSeconds);
27. this.Score.Text = Shared.Mechanics.GameScoreStr;
28. this.Coins.Text = string.Format("\*{0:00}", Shared.Mechanics.CoinsCount);
29. this.Lives.Text = string.Format("\*{0:00}", Shared.Mechanics.Lives);
30. }
31. protected override void Update()
32. {
33. this.Time.Text = string.Format("{0:000}", Shared.Mechanics.TimeLeft.TotalSeconds);
34. this.Score.Text = Shared.Mechanics.GameScoreStr;
35. this.Coins.Text = string.Format("\*{0:00}", Shared.Mechanics.CoinsCount);
36. this.Lives.Text = string.Format("\*{0:00}", Shared.Mechanics.Lives);
37. }
38. }
39. }

#### Assets/Scripts/MainMenuSpawnScript.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models;
4. namespace MarIO.Assets.Scripts
5. {
6. public class MainMenuSpawnScript : Script
7. {
8. private Vector3 Position;
9. public MainMenuSpawnScript(GameObject Parent) : base(Parent)
10. { }
11. protected override void OnColliderEnter(Collider e)
12. {
13. e.Parent.Transform.Position = Position;
14. }
15. protected override void Start()
16. {
17. Position = new Vector3(320, 176, 0);
18. Goomba e = new Goomba()
19. {
20. Name = "Bot"
21. };
22. e.Transform.Position = Position;
23. }
24. protected override void Update()
25. { }
26. }
27. }

#### Assets/Scripts/MarioTriggerColliderScript.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models;
4. using MarIO.Assets.Models.Miscellaneous;
5. using System.Diagnostics;
6. namespace MarIO.Assets.Scripts
7. {
8. public class BottomMarioChecker : Script
9. {
10. private Mario Mario;
11. public BottomMarioChecker(GameObject Parent) : base(Parent)
12. { }
13. protected override void OnColliderEnter(Collider e)
14. {
15. if (e.Parent is Enemy)
16. {
17. Enemy tmp = e.Parent as Enemy;
18. Debug.WriteLine(string.Format("Zabil jsi {0}", tmp.Name));
19. tmp.IsDestroyed = true;
20. Mario.KilledEnemy = true;
21. }
22. else if (e.Parent is PowerUp)
23. {
24. ((PowerUp)e.Parent).OnPickedUp?.Invoke();
25. }
26. }
27. protected override void Start()
28. {
29. Mario = GameObject.Find<Mario>("Player");
30. }
31. protected override void Update()
32. { }
33. }
34. internal class TopMarioChecker : Script
35. {
36. private Mario Mario;
37. public TopMarioChecker(GameObject Parent) : base(Parent)
38. { }
39. protected override void OnColliderEnter(Collider e)
40. {
41. if (e.Parent is Enemy)
42. {
43. Debug.WriteLine(string.Format("Zabilo Tě {0}", e.Parent.TypeName));
44. Mario.CurrentState--;
45. }
46. else if (e.Parent is Block)
47. {
48. Block tmp = e.Parent as Block;
49. if (tmp.State == Block.CollisionState.Stay)
50. {
51. tmp.AnimateBlockCollision();
52. }
53. tmp.GetContent();
54. }
55. else if (e.Parent is PowerUp)
56. {
57. ((PowerUp)e.Parent).OnPickedUp?.Invoke();
58. }
59. }
60. protected override void Start()
61. {
62. Mario = GameObject.Find<Mario>("Player");
63. }
64. protected override void Update()
65. { }
66. }
67. internal class LeftMarioChecker : Script
68. {
69. private Mario Mario;
70. public LeftMarioChecker(GameObject Parent) : base(Parent)
71. { }
72. protected override void OnColliderEnter(Collider e)
73. {
74. if (e.Parent is Enemy)
75. {
76. Debug.WriteLine(string.Format("Zabilo Tě {0}", e.Parent.TypeName));
77. Mario.CurrentState--;
78. //Mario?.Destroy();
79. }
80. else if (e.Parent is PowerUp)
81. {
82. ((PowerUp)e.Parent).OnPickedUp?.Invoke();
83. }
84. }
85. protected override void Start()
86. {
87. Mario = GameObject.Find<Mario>("Player");
88. }
89. protected override void Update()
90. { }
91. }
92. internal class RightMarioChecker : Script
93. {
94. private Mario Mario;
95. public RightMarioChecker(GameObject Parent) : base(Parent)
96. { }
97. protected override void OnColliderEnter(Collider e)
98. {
99. if (e.Parent is Enemy)
100. {
101. Debug.WriteLine(string.Format("Zabilo Tě {0}", e.Parent.TypeName));
102. Mario.CurrentState--;
103. //Mario?.Destroy();
104. }
105. else if (e.Parent is PowerUp)
106. {
107. ((PowerUp)e.Parent).OnPickedUp?.Invoke();
108. }
109. }
110. protected override void Start()
111. {
112. Mario = GameObject.Find<Mario>("Player");
113. }
114. protected override void Update()
115. { }
116. }
117. }

#### Assets/Scripts/MusicScript.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using System;
4. using System.Diagnostics;
5. namespace MarIO.Assets.Scripts
6. {
7. public class MusicScript : Script
8. {
9. private Sound Music;
10. private TimeSpan MusicLenght;
11. private Stopwatch Timer;
12. public MusicScript(GameObject Parent) : base(Parent)
13. { }
14. protected override void OnColliderEnter(Collider e)
15. { }
16. protected override void Start()
17. {
18. Music = Shared.Assets.Sounds.OVERWORLD\_THEME\_SOUND;
19. MusicLenght = Music.FileReader.TotalTime;
20. Shared.Mechanics.FXSoundSource.PlaySound(Music);
21. Timer = Stopwatch.StartNew();
22. }
23. protected override void Update()
24. {
25. if (Timer.Elapsed > MusicLenght)
26. {
27. Shared.Mechanics.FXSoundSource.PlaySound(Music);
28. Timer.Restart();
29. }
30. }
31. }
32. }

#### Assets/Scripts/PipePort.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models;
4. namespace MarIO.Assets.Scripts
5. {
6. public class PipePort : Script
7. {
8. private Mario Player;
9. public Block Pipe;
10. public PipePort(GameObject Parent) : base(Parent)
11. { }
12. protected override void OnColliderEnter(Collider e)
13. {
14. if (Pipe.SpecialAction != null)
15. {
16. if (e.Parent == Player)
17. {
18. switch (Pipe.PipeEnterDirection)
19. {
20. case Transform.Direction.Up:
21. break;
22. case Transform.Direction.Left:
23. break;
24. case Transform.Direction.Down:
25. if (Player.CurrentMovement == Mario.Movement.Crouching)
26. {
27. Player.PipeEnter(Pipe);
28. }
29. break;
30. case Transform.Direction.Right:
31. if (Player.CurrentMovement == Mario.Movement.Standing)
32. {
33. Player.PipeEnter(Pipe);
34. }
35. break;
36. default:
37. break;
38. }
39. }
40. }
41. }
42. protected override void Start()
43. {
44. Player = GameObject.Find<Mario>("Player");
45. Pipe = (Block)Parent;
46. }
47. protected override void Update()
48. { }
49. }
50. }

#### Assets/Scripts/PowerUpScript.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using MarIO.Assets.Models;
5. using MarIO.Assets.Models.Miscellaneous;
6. using System;
7. using static DKEngine.Core.Components.Transform;
8. namespace MarIO.Assets.Scripts
9. {
10. internal class PowerUpScript : Script
11. {
12. private PowerUp Target;
13. private bool CreatedForFirstTime = true;
14. private bool CreatedAnimation = true;
15. private float CreatedStartY;
16. private const float CreationAnimationSpeed = 20f;
17. private const float Speed = 80f;
18. private const float FloatSpeed = 250f;
19. private const float Acceleration = 3.5f;
20. private float CurrentSpeed = 0;
21. private float vertSpeed = 0;
22. private bool IsFalling = false;
23. private bool Jumped = false;
24. public PowerUpScript(GameObject Parent) : base(Parent)
25. {
26. Target = Parent as PowerUp;
27. }
28. protected override void OnColliderEnter(Collider e)
29. { }
30. protected override void Start()
31. {
32. CurrentSpeed = Speed;
33. Target.PlayerReference = GameObject.Find<Mario>("Player");
34. }
35. protected override void Update()
36. {
37. if (CreatedForFirstTime)
38. {
39. Target.Collider.Enabled = false;
40. CreatedStartY = Target.Transform.Position.Y;
41. CreatedForFirstTime = false;
42. return;
43. }
44. else if (CreatedAnimation)
45. {
46. if (CreatedStartY < Target.Transform.Position.Y + 16)
47. {
48. Target.Transform.Position -= new Vector3(0, Engine.DeltaTime \* CreationAnimationSpeed, 0);
49. }
50. else
51. {
52. Target.Transform.Position = new Vector3(Target.Transform.Position.X, CreatedStartY - 16, Target.Transform.Position.Z);
53. Target.Collider.Enabled = true;
54. CreatedAnimation = false;
55. }
56. return;
57. }
58. else
59. {
60. switch (Target.Type)
61. {
62. case PowerUp.PowerUpType.Mushroom:
63. MushroomMovement();
64. break;
65. case PowerUp.PowerUpType.Flower:
66. CurrentSpeed = 0;
67. break;
68. case PowerUp.PowerUpType.Star:
69. StarMovement();
70. break;
71. default:
72. throw new Exception("JAK");
73. }
74. }
75. }
76. private void MushroomMovement()
77. {
78. if (Target.Collider.Collision(Direction.Left))
79. {
80. CurrentSpeed = Speed;
81. }
82. if (Target.Collider.Collision(Direction.Right))
83. {
84. CurrentSpeed = -Speed;
85. }
86. if (!Target.Collider.Collision(Direction.Down))
87. {
88. if (!IsFalling)
89. {
90. vertSpeed = 0;
91. IsFalling = true;
92. }
93. else
94. {
95. if (vertSpeed < FloatSpeed)
96. {
97. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed;
98. }
99. else
100. {
101. vertSpeed = FloatSpeed;
102. }
103. }
104. }
105. else if (IsFalling)
106. {
107. vertSpeed = 0;
108. IsFalling = false;
109. }
110. Target.Transform.Position += new Vector3(CurrentSpeed \* Engine.DeltaTime, vertSpeed \* Engine.DeltaTime, 0);
111. }
112. private void StarMovement()
113. {
114. if (Target.Collider.Collision(Direction.Left))
115. {
116. CurrentSpeed = Speed;
117. }
118. if (Target.Collider.Collision(Direction.Right))
119. {
120. CurrentSpeed = -Speed;
121. }
122. if (!Target.Collider.Collision(Direction.Down))
123. {
124. if (vertSpeed == 0 && !Jumped)
125. {
126. vertSpeed = -FloatSpeed \* 1.5f;
127. Jumped = true;
128. }
129. else if (!Target.Collider.Collision(Direction.Up) && vertSpeed < 0)
130. {
131. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed;
132. }
133. else
134. {
135. vertSpeed = 0;
136. IsFalling = true;
137. }
138. }
139. else
140. {
141. if (!IsFalling && !Jumped)
142. {
143. vertSpeed = 0;
144. Jumped = true;
145. IsFalling = true;
146. }
147. else if (IsFalling)
148. {
149. if (vertSpeed < FloatSpeed)
150. {
151. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed;
152. }
153. else
154. {
155. vertSpeed = FloatSpeed;
156. }
157. }
158. }
159. }
160. }
161. }

#### Assets/Scripts/SpecialBlocksUpdateScript.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. namespace MarIO.Assets.Scripts
4. {
5. public class SpecialBlocksUpdateScript : Script
6. {
7. public SpecialBlocksUpdateScript(GameObject Parent)
8. : base(Parent)
9. { }
10. protected override void OnColliderEnter(Collider e)
11. { }
12. protected override void Start()
13. {
14. }
15. protected override void Update()
16. {
17. while (Shared.AnimatedWorldReferences.SpecialActions.Count > 0)
18. {
19. Shared.AnimatedWorldReferences.SpecialActions.Pop().SpecialAction();
20. }
21. }
22. }
23. }

#### Assets/Scripts/WorldChangeManagerScript.cs

1. using DKEngine.Core;
2. using DKEngine.Core.Components;
3. using MarIO.Assets.Models;
4. namespace MarIO.Assets.Scripts
5. {
6. public class WorldChangeManagerScript : Script
7. {
8. public Block CurrentlyEnteredPipeScript;
9. public WorldChangeManagerScript(GameObject Parent) : base(Parent)
10. {
11. Name = "worldManager";
12. }
13. protected override void OnColliderEnter(Collider e)
14. { }
15. protected override void Start()
16. { }
17. protected override void Update()
18. {
19. CurrentlyEnteredPipeScript?.SpecialAction();
20. CurrentlyEnteredPipeScript = null;
21. }
22. }
23. }

#### Assets/Scripts/WorldEnd.cs

1. using DKEngine;
2. using DKEngine.Core;
3. using DKEngine.Core.Components;
4. using MarIO.Assets.Models;
5. using MarIO.Assets.Scenes;
6. using System;
7. using static DKEngine.Core.Components.Transform;
8. namespace MarIO.Assets.Scripts
9. {
10. internal class WorldEnd : Script
11. {
12. private Mario Player;
13. private CharacterController PlayerController;
14. private Animator PlayerAnimator;
15. private float horiSpeed = 0;
16. private float vertSpeed = 0;
17. private float Distance = 180;
18. private float startX;
19. private const float MovementSpeed = 80f;
20. private const float FloatSpeed = 300f;
21. private const float Acceleration = 3.5f;
22. private readonly TimeSpan \_delay = new TimeSpan(0, 0, 3);
23. private TimeSpan Delay = new TimeSpan();
24. private string MOVE
25. {
26. get
27. {
28. switch (Player.CurrentState)
29. {
30. case Mario.State.Small:
31. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_MOVE\_LEFT;
32. case Mario.State.Super:
33. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_SUPER\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_SUPER\_MOVE\_LEFT;
34. case Mario.State.Fire:
35. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_FIRE\_MOVE\_LEFT;
36. /\*case Mario.State.Invincible:
37. return horiSpeed >= 0 ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_MOVE\_RIGHT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_MOVE\_LEFT;\*/
38. default:
39. throw new Exception("JAK");
40. }
41. }
42. }
43. private string IDLE
44. {
45. get
46. {
47. switch (Player.CurrentState)
48. {
49. case Mario.State.Dead:
50. case Mario.State.Small:
51. return horiSpeed < 0 ? Shared.Assets.Animations.MARIO\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_IDLE\_RIGHT;
52. case Mario.State.Super:
53. return horiSpeed < 0 ? Shared.Assets.Animations.MARIO\_SUPER\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_SUPER\_IDLE\_RIGHT;
54. case Mario.State.Fire:
55. return horiSpeed < 0 ? Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_FIRE\_IDLE\_RIGHT;
56. /\*case Mario.State.Invincible:
57. return IsFacingLeft ? Shared.Assets.Animations.MARIO\_INVINCIBLE\_IDLE\_LEFT : Shared.Assets.Animations.MARIO\_INVINCIBLE\_IDLE\_RIGHT;\*/
58. default:
59. throw new Exception("JAK");
60. }
61. }
62. }
63. public WorldEnd(GameObject Parent) : base(Parent)
64. {
65. startX = Parent.Transform.Position.X;
66. }
67. protected override void OnColliderEnter(Collider e)
68. {
69. if (e.Parent is Mario)
70. PlayerController.Enabled = false;
71. }
72. protected override void Start()
73. {
74. Player = GameObject.Find<Mario>("Player");
75. PlayerAnimator = Component.Find<Animator>("Player\_Animator");
76. PlayerController = Script.Find<CharacterController>(nameof(CharacterController));
77. }
78. protected override void Update()
79. {
80. if (Shared.Mechanics.TimeLeft.TotalSeconds <= 0)
81. {
82. Player.CurrentState = Mario.State.Dead;
83. Shared.Mechanics.TimeCounter.Stop();
84. }
85. if (!PlayerController.Enabled)
86. {
87. PlayerAnimator.Play(MOVE);
88. if (!Player.Collider.Collision(Direction.Down))
89. {
90. if (vertSpeed < FloatSpeed)
91. {
92. vertSpeed += Engine.DeltaTime \* Acceleration \* FloatSpeed;
93. if (vertSpeed > FloatSpeed)
94. vertSpeed = FloatSpeed;
95. }
96. else
97. {
98. vertSpeed = FloatSpeed;
99. }
100. }
101. if (Player.Transform.Position.X > startX + Distance)
102. {
103. PlayerAnimator.Play(IDLE);
104. Delay += new TimeSpan(0, 0, 0, 0, (int)(Engine.DeltaTime \* 1000));
105. if (\_delay < Delay)
106. {
107. Shared.Mechanics.FXSoundSource.StopSound(Shared.Assets.Sounds.OVERWORLD\_THEME\_SOUND);
108. Engine.ChangeScene(nameof(GameOver), true);
109. }
110. }
111. else
112. {
113. this.Player.Transform.Position += new Vector3(MovementSpeed \* Engine.DeltaTime, vertSpeed, 0);
114. }
115. }
116. }
117. }
118. }

# Závěr

Celý kód ročníkové práce je dostupný na mém osobím GitHubu, který jste si mohli přečíst v kapitole „Úvod“. Program pracuje s mnoha různými strukturami, třídami, metodami, ať už těch napsaných mnou v jazyku C# nebo různých dalších naimporotvaných C++ knihoven. V základu se jedna o velice jednoduchý 2D herní engine pracující se slušnou vykonností. Zvládá přehrávat animace pomocí obrázků gif, vakreslovat objekty přes sebe s průhledností nebo přehrávat zvukové efekty. Využito bylo různých knihoven, se kterými jsem měl možnost se naučit spousty nových dovedností. Jednou z nich byla knihovna NAudio, sloužící k přehrávání zvuků. O té zde byla zmínka v úvodu této ročníkové práce. Program se povedlo dostat do prezentovatelné podoby a to díky podpoře mých přátel. Tímto bych chtěl poděkovat mým kamarádům, Marianu Dolinskému, Tomáši Lošťákovi a Pavlu Jakubcovi za jejich podporu.

# Literatura

MSDN [on-line] [cit. 2017/05/14] Dostupné na Microsoft Developer Network  
https://msdn.microsoft.com/cs-cz/default.aspx

[StackOverFlow](http://www.StackOverFlow.com) [on-line] [cit. 2017/05/14] Dostupné na StackOverFlow  
<https://stackoverflow.com/>

Přílohy

1. Stromová struktura solution