Duarte Maia
Work Snapshot

dsalgadomaia@gmail.com
http://duartemaia.com

Included Works

- **Driftking** – Gameplay Programming *is* Design
- **Simpleton** – Tooling for the non-technical, Systems Design

Simpleton

dsalgadomaia@gmail.com
http://duartemaia.com

DriftKing

- Gameplay Programming *is* Design -

DRIFTKING

Overview

dsalgadomaia@gmail.com
http://duartemaia.com

About the Project

3v3 Multiplayer Online Car Battle

Designer / Gameplay Programmer

3 months, 6-man team

Made in Unity, coded in C#

About the Game

2 Available Cars w/ asymmetrical abilities

Kill the King / Deathmatch Game Mode

Drift to Cast Spells

DRIFTKING

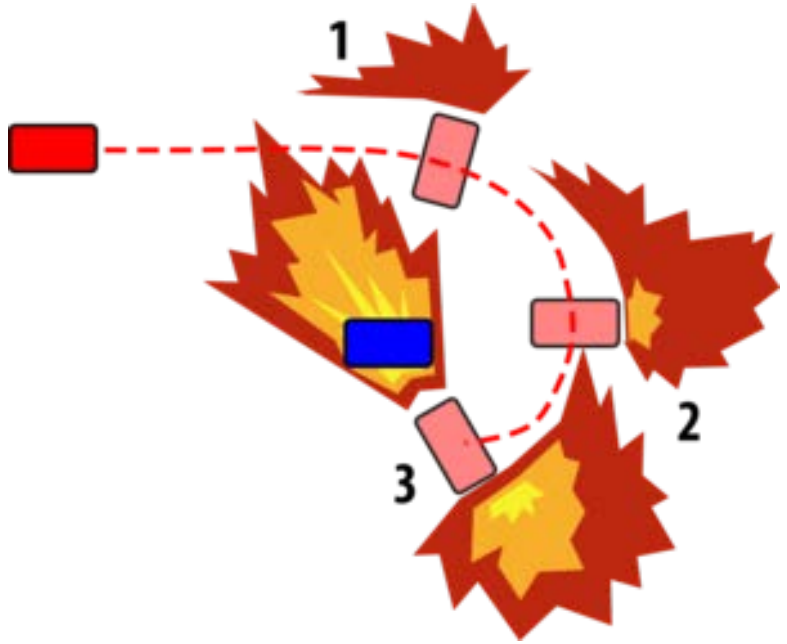
Gameplay Trailer

dsalgadomaia@gmail.com
http://duartemaia.com

DRIFTKING

Core Loop Overview

dsalgadomaia@gmail.com
http://duartemaia.com



In the diagram to the left:

1. The red car casts a fire spell at the blue car
2. The red car enters a drift. As the drift progresses, the spell build-up increases, amping damage, range, and other per-spell parameters
3. After the drift is done, the red car casts a flamethrower spell directly in front of it, towards the blue car

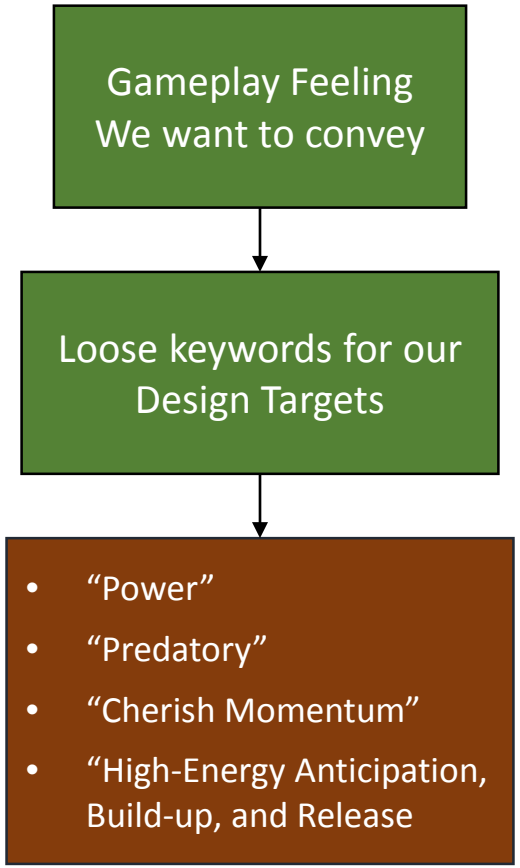
dsalgadomaia@gmail.com
http://duartemaia.com

Design Process

DRIFTKING

Design Process

dsalgadomaia@gmail.com
http://duartemaia.com



Resulting Design Pillars

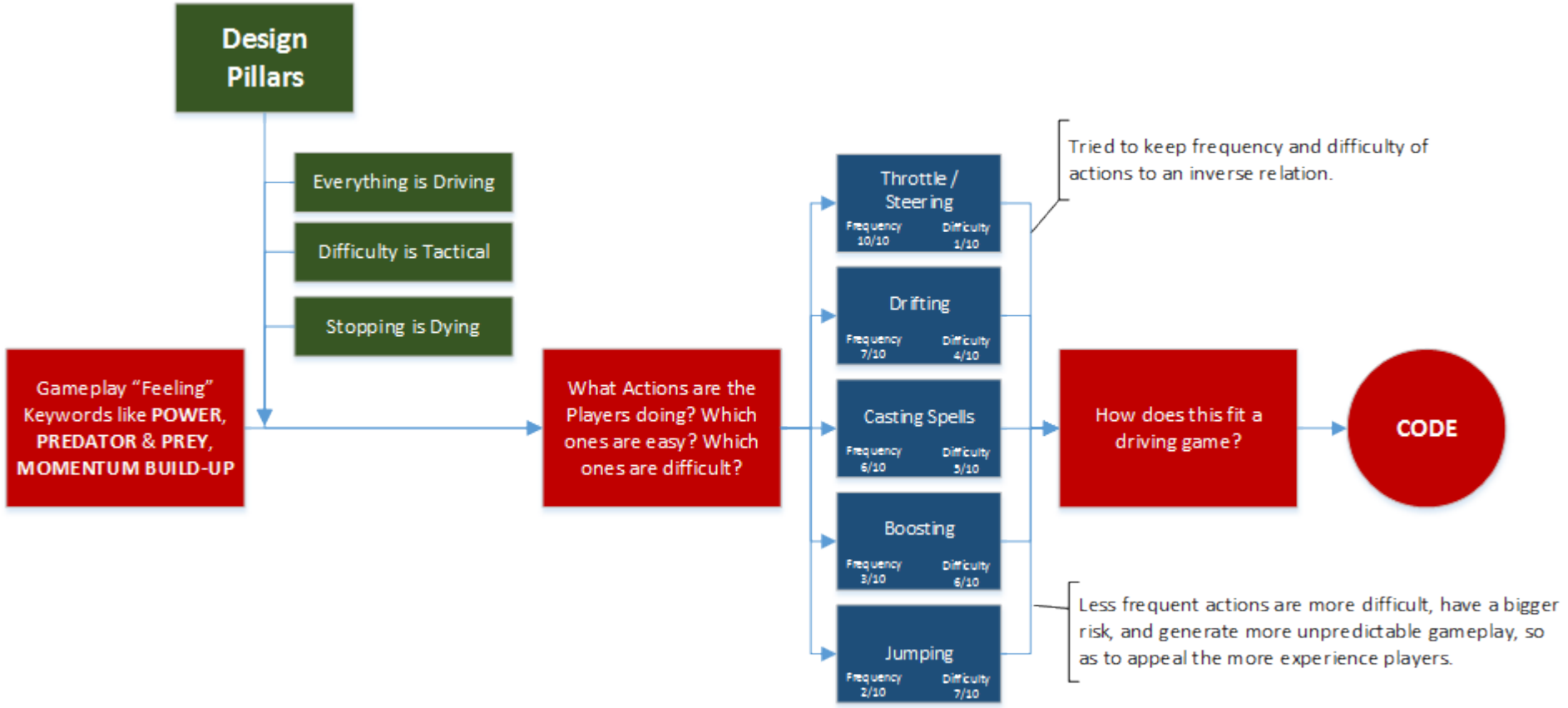
Everything is Driving	Car Combat games almost always use Twin-stick shooting. One stick controls the car, the other controls the aiming. This means that players doing one or the other successfully, never both. The idea of tying the combat with the actual driving past the dichotomy appealed to us
Difficulty is Tactical	Since our target market demanded a short, polished experience, we wanted to focus on making the game playable for new players. This meant veering away from dexterity-based difficulty to a more beginner-friendly game with a tactical mastery
Stopping is Dying	We wanted out players to cherish their momentum and feel in full all of the stages of their interaction (decision, build-up, release, retreat). So, we wanted to have the players feel the danger of leaving themselves stopped, with no chance of building up a drift or a retreat

We refered back to our Design Pillars when making decisions about the design of the game. Features/Tweaks that went against our pillars were largely discarded, with few pondered exceptions

DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Translating Pillars to Features



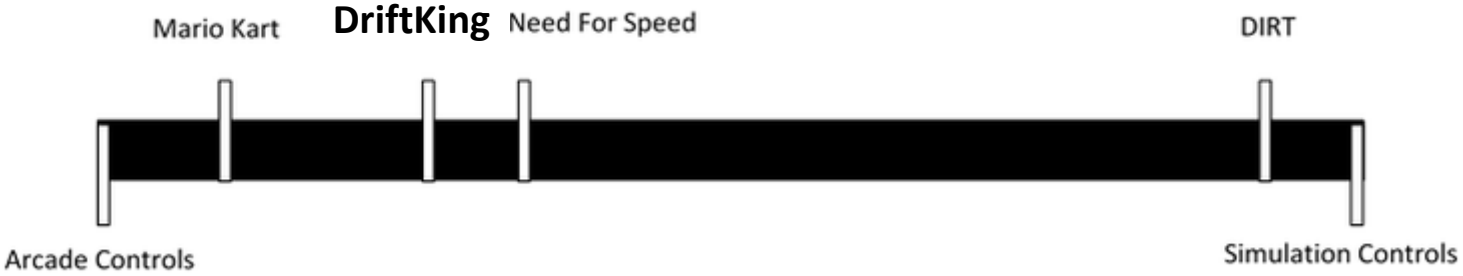
DRIFTKING

Competitive Benchmarking

dsalgadomaia@gmail.com
http://duartemaia.com

Where do we fit in the market?

Arcade / twitch controls



Simulation controls

DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Controls

- Gameplay Programming *is* Design -

DRIFTKING

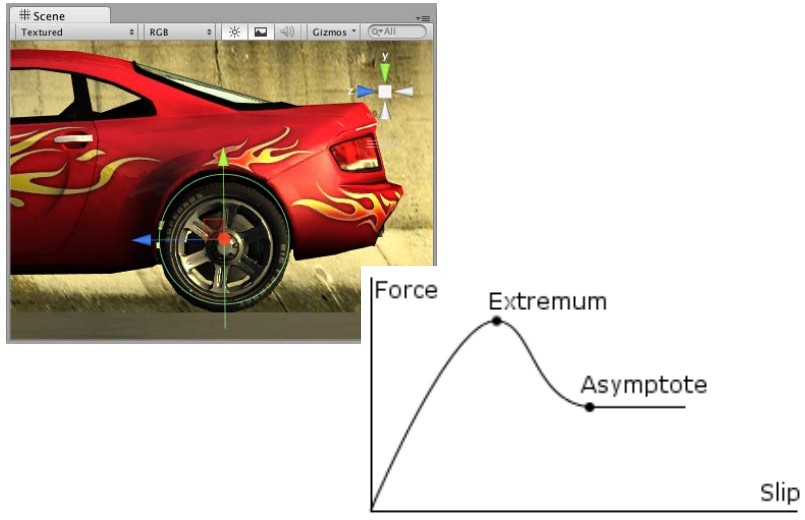
Gameplay Programming *is* Design

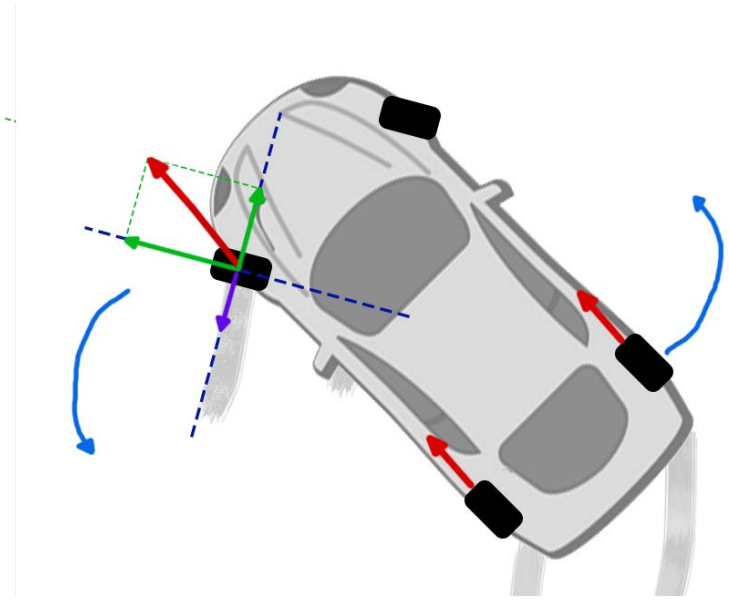
dsalgadomaia@gmail.com
http://duartemaia.com

We decided to start out from a very realistic simulation physics system.

This ensured that the experience would remain relatable as we progressively broke away from that realism.

Having started with a 4-point spring system with sideways & forward friction on the wheels, motor torque, and tyre slippage extremums, we progressively added input-dependent forces and torques to drive player behaviour towards our Design Targets





dsalgadomaia@gmail.com
http://duartemaia.com

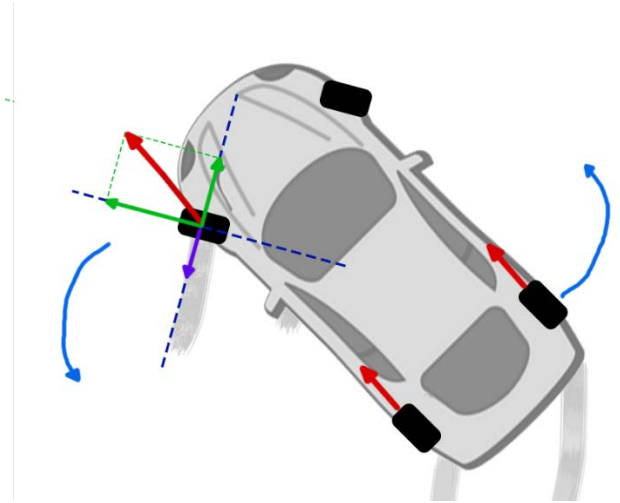
Breaking the realism

While **Driving**:

- We clamp the steering angle while the player is not drifting so they cannot go into a drift accidentally, and so players are enticed to drift to make sharper turns

And while **Drifting**:

- We apply a **forward-facing force** on the back wheels while drifting that is dependent on the acceleration input (diagram below)
- We apply a **Torque** directly on the car (bypassing the wheel collider physics) that is dependent on the steering input
- We apply an **Impulse Torque** for the first steering input, so the car snaps its rotation to a slight sideways trajectory
- We apply a **counter-steer amplification factor** to the steering Torque, controlling how easy or difficult it is to counter-steer a drift



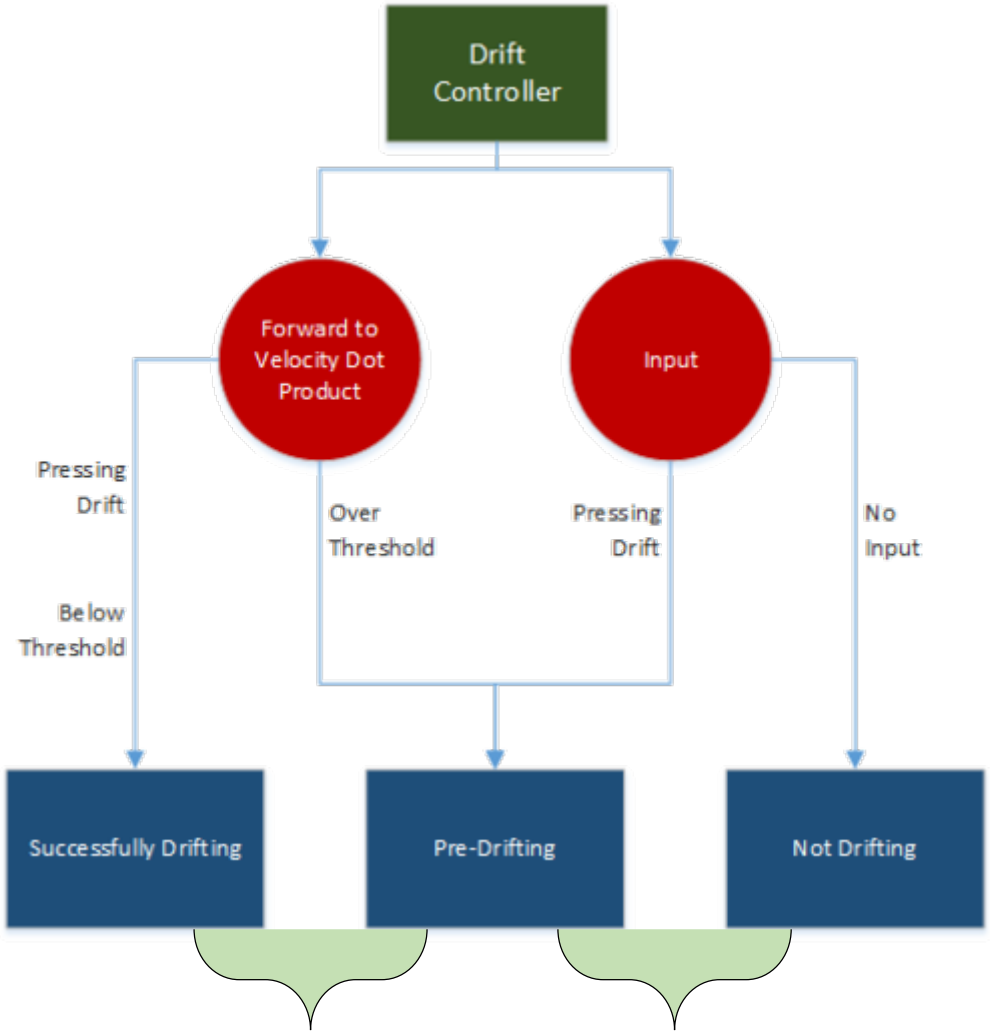
DRIFTKING

Gameplay Programming *is* Design

The result was a tri-state machine for Drifting

The tricky part is to make these gameplay states transition smoothly, which was achieved through non-linear amortizations of values within the transition threshold

This helped us make sure that we could develop each state individually, with very separate design targets that only blend into each other



Amortization of values within a given transition-bound threshold gently faded the added arcade forces

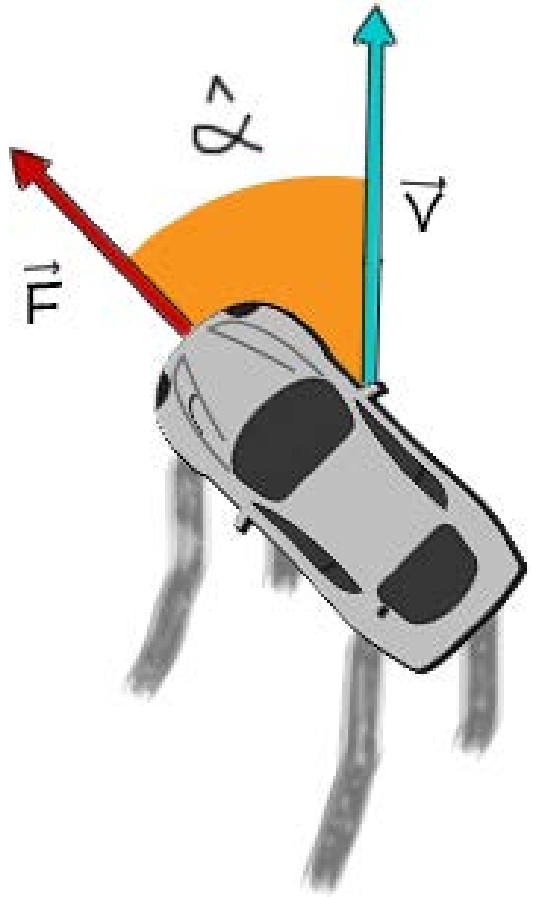
DRIFTKING

Gameplay Programming *is* Design

dsalgadomaia@gmail.com
http://duartemaia.com

So how does Drifting equate to spellcasting?

Drift Quality dictates the success of your spells:
Better drifts make for more powerful, far-reaching, and precise spellcasting



F *-> Normalized Forward-Facing Vector*
V *-> Normalized Velocity Bearing*
α *-> Angle between F and V*

The drift quality is computed by factoring the angle between the velocity and forward, and also the total distance travelled (ΔX). Frame by frame, this is the equation that gives us the drift quality, if we assume a magnitude M.

$$\Delta DQ = |DOT(F,V)| * \Delta X * M$$
$$DQ = SUM(\Delta DQ)$$

DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Spellcasting Architecture

- Empowering explorative design -

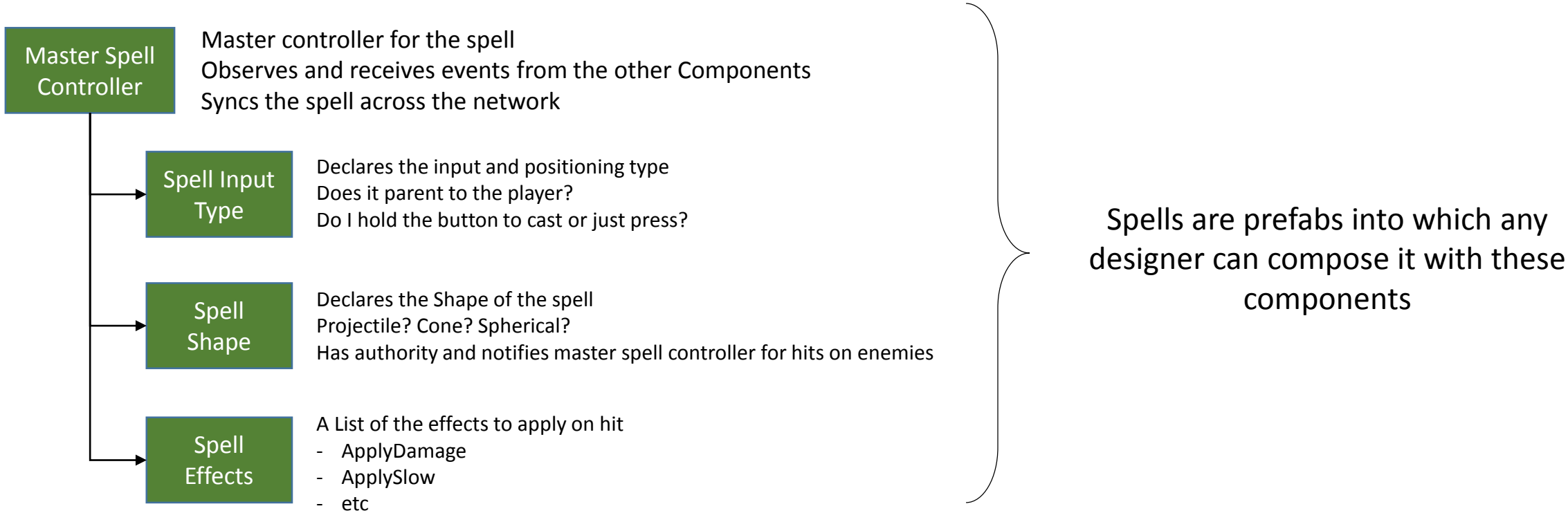
DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Spellcasting

I wanted to allow for experimentation with spell design.
I knew we could whiteboard for a thousand years and still not know whether any of it was fun.

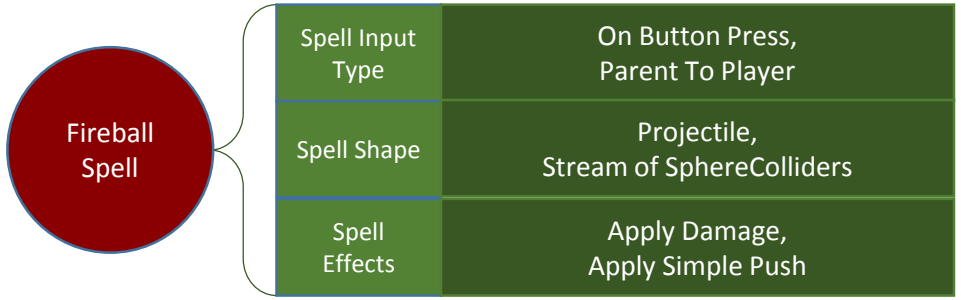
I decided to create a modular spell architecture to empower experimentation

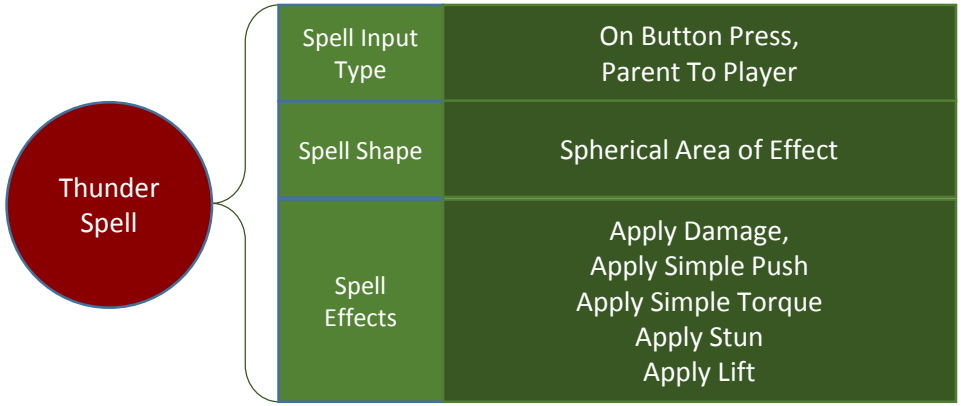


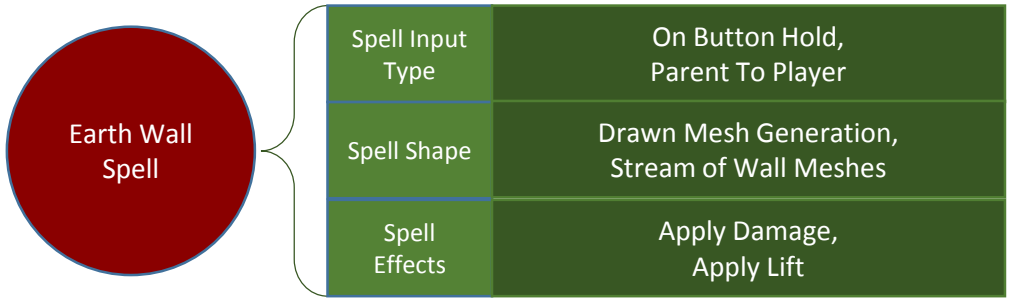
DRIFTKING

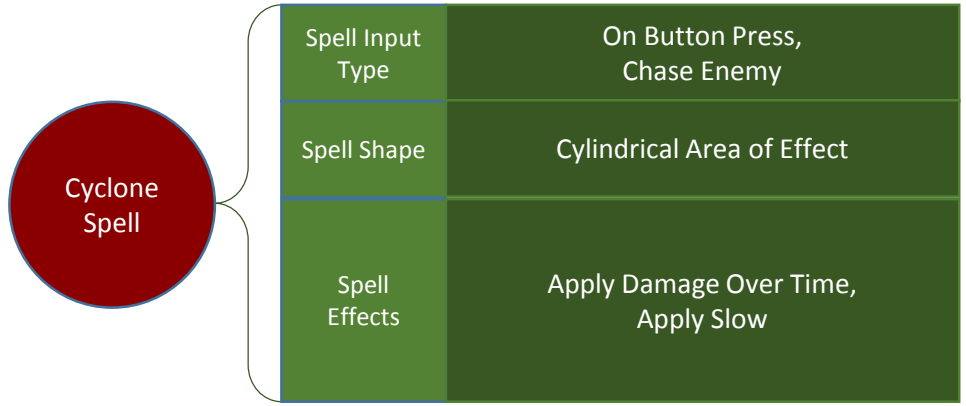
Spellcasting

dsalgadomaia@gmail.com
http://duartemaia.com









Spellcasting

dsalgadomaia@gmail.com
http://duartemaia.com

This architecture allowed me to rapidly prototype new spell effects. By decoupling the protocol that detects a hit with the protocols that declare what to do with it, we were able to freely test new ideas

Of course, many Spell Effects did not make it into the final game, such as applying concentric forces, jumbled controls, camera effects, and many other disparate ideas

DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Code – DriftController State Machine

GitHub Link

```
#region Components References

    private SteeringControl SteerController;
    private SpellCasting SpellCastingController;
    private Rigidbody Rb;

    private Image hudDrift;
    private Toggle devToggle;

#endregion


#region Arcadey-feel Applied Forces Variables

    [Header("Arcadey-feel Applied Forces")]
    [SerializeField] private Transform backWheelsForcePoint;
    [SerializeField] private float passiveDriftForce = 1000;
    [SerializeField] private float activeDriftSteerTorque = 3000;
    [SerializeField] private float activeDriftCounterSteerTorque = 25000;
    [SerializeField] private float EndDriftImpulse = 5000;
    [SerializeField] private float throttleDriftForceForward = 2000;
    [SerializeField] private float throttleDriftTorque = 2000;
    [SerializeField] private float throttleDriftTorqueCutoffSpeed = 150;

#endregion
```

```
#region Pre-drift Variables

    [Header("Pre-drift variables")]
    [SerializeField] private float SuccessfulDriftMaxDot = 0.83f;
    [SerializeField] private float SuccessfulDriftMinDot = - 0.7f;
    [SerializeField] private float PreDriftRotationSpeed = 10f;

#endregion

#region Drift Mechanic Variables

    [Header("Drift Mechanic Variables")]
    [SerializeField] private float MinSpeedToDrift = 50f;
    [SerializeField] private float MinSteerHelperValueWhileDrifting = 0.43f;
    [SerializeField] private float MaxSteerHelperValueWhileDrifting = 0.55f;
    [SerializeField] private float TransferSidewaysMomentumToForwardTimeDuration =
0.8f;
    [HideInInspector] public Drift CurrentDrift;
    private Coroutine SidewaysMomentumTransferCoroutine;
    private bool SidewaysMomentumTransferCoroutineIsRunning;
    private int DriftPercentageInt;

#endregion
```

```
#region State Monitoring

    [HideInInspector] public bool isPressingDriftButton;

    public DriftState CurrentDriftState { get; private set; }
    public enum DriftState
    {
        NotDrifting,
        PreDrifting,
        DriftingSuccessfully
    }

#endregion
```

DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Code – Support Class Drift

GitHub Link

```
public class Drift
{
    public bool isValid;
    public float driftDirection;
    public float distanceTravelledDelta;
    public float driftQualityDelta;
    public float forwardToVelocityDot;

    public Drift(bool is_valid)
    {
        isValid = is_valid;
        driftDirection = 0f;
        distanceTravelledDelta = 0f;
        driftQualityDelta = 0f;
        forwardToVelocityDot = 1f;
    }

    public void UpdateDriftValues(Transform carTransform, Rigidbody carRb, float steer, float throttle,
                                  float driftPowerAngleContributionMagnitude,
                                  float driftPowerDistanceContributionMagnitude)
    {
        forwardToVelocityDot = Vector3.Dot(carTransform.forward, carRb.velocity.normalized);
        driftQualityDelta = CalculateFrameDriftPower(carRb, forwardToVelocityDot,
                                                     driftPowerAngleContributionMagnitude,
                                                     driftPowerDistanceContributionMagnitude);
        driftDirection = Mathf.Sign(
                            Vector3.Cross(
                                HelperFunctions.NullifyVectorY(carRb.velocity.normalized),
                                HelperFunctions.NullifyVectorY(carTransform.forward)).y *
                                    (1 - Mathf.Abs(forwardToVelocityDot)
                                )
                            );
    }
}
```

```
public class Drift
 {

    private float CalculateFrameDriftPower(Rigidbody carRb, float forwardToVelocityDot,
                                           float driftPowerAngleContributionMagnitude,
                                           float driftPowerDistanceContributionMagnitude)
    {
        if (forwardToVelocityDot < -0.5f)
            return 0f;

        //Got this value from testing.
        //There really is no other way to parametrize distance travelled into values between 0 and 1
        float MaxDistanceTravelledDelta = 13f;

        float DriftAngleDelta = (1 - Mathf.Clamp(forwardToVelocityDot, 0f, 1f));
        float DistanceTravelledDelta = Mathf.Clamp((carRb.velocity.magnitude * Time.fixedDeltaTime), 0f, MaxDistanceTravelledDelta);
        float DistanceTravelledDeltaNormalized = DistanceTravelledDelta / MaxDistanceTravelledDelta;

        float driftPowerDelta = (driftPowerAngleContributionMagnitude * DriftAngleDelta)
                                + (DistanceTravelledDeltaNormalized * driftPowerDistanceContributionMagnitude);

        return driftPowerDelta;
    }
}
```

DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Code – DriftController Update

GitHub Link

```
private void UpdateSuccessfulDrift(float steer, float throttle)
{
    ApplySuccessfulDriftForces(steer, throttle);

    if (!DidCastThisDrift)
      BuildDriftPower();
}


public void ApplySuccessfulDriftForces(float steer, float throttle)
{
    ApplySteerTorque(steer);
    ApplyThrottleTorque(throttle);
    ApplyPassiveDriftForce();
    ApplyThrottleMomentum(throttle);

    float parametrizedDot = SuccessfulDriftMaxDot - CurrentDrift.forwardToVelocityDot;
    float steerHelperValue = MinSteerHelperValueWhileDrifting + parametrizedDot
                             * (MaxSteerHelperValueWhileDrifting - MinSteerHelperValueWhileDrifting);

    SteerController.ToggleSteerHelper(steerHelperValue, 0f);
}
```

```
private void ApplyPreDriftForces(float steerInput, float throttleInput)
  {
    if (throttleInput == 0f || steerInput == 0f || SteerController.CurrentSpeed <= 10
      || Mathf.Abs(Rb.angularVelocity.y) > PreDriftRotationSpeed)
          return;

    float rawAngularSpeed = PreDriftRotationSpeed;

    float dotProductFactor = Mathf.Max(
                          1f - (1f - CurrentDrift.forwardToVelocityDot) / (1f - SuccessfulDriftMaxDot),
                          0.5f);

    float speedFactor = Mathf.Clamp(HelperFunctions.CurveFactor(SteerController.CurrentSpeed
                                                    / SteerController.OriginalTopspeed), 0f, 1f);
    float angularSpeed = rawAngularSpeed * dotProductFactor * speedFactor;


    if (Mathf.Sign(CurrentDrift.driftDirection) == -Mathf.Sign(steerInput))
      angularSpeed /= 1.3f;


    Rb.angularVelocity = new Vector3(Rb.angularVelocity.x, steerInput * angularSpeed,  Rb.angularVelocity.z);

    ApplySteerTorque(steerInput * 0.3f);
}
```

dsalgadomaia@gmail.com
http://duartemaia.com

Simpleton

- Tooling for the non-technical, systems design -

Simpleton

Overview

dsalgadomaia@gmail.com
http://duartemaia.com

About the Project

Mobile Point & Click Narrative Adventures

Designer / Solo Programmer

Made in Unity, coded in C#

In Development

About the Game

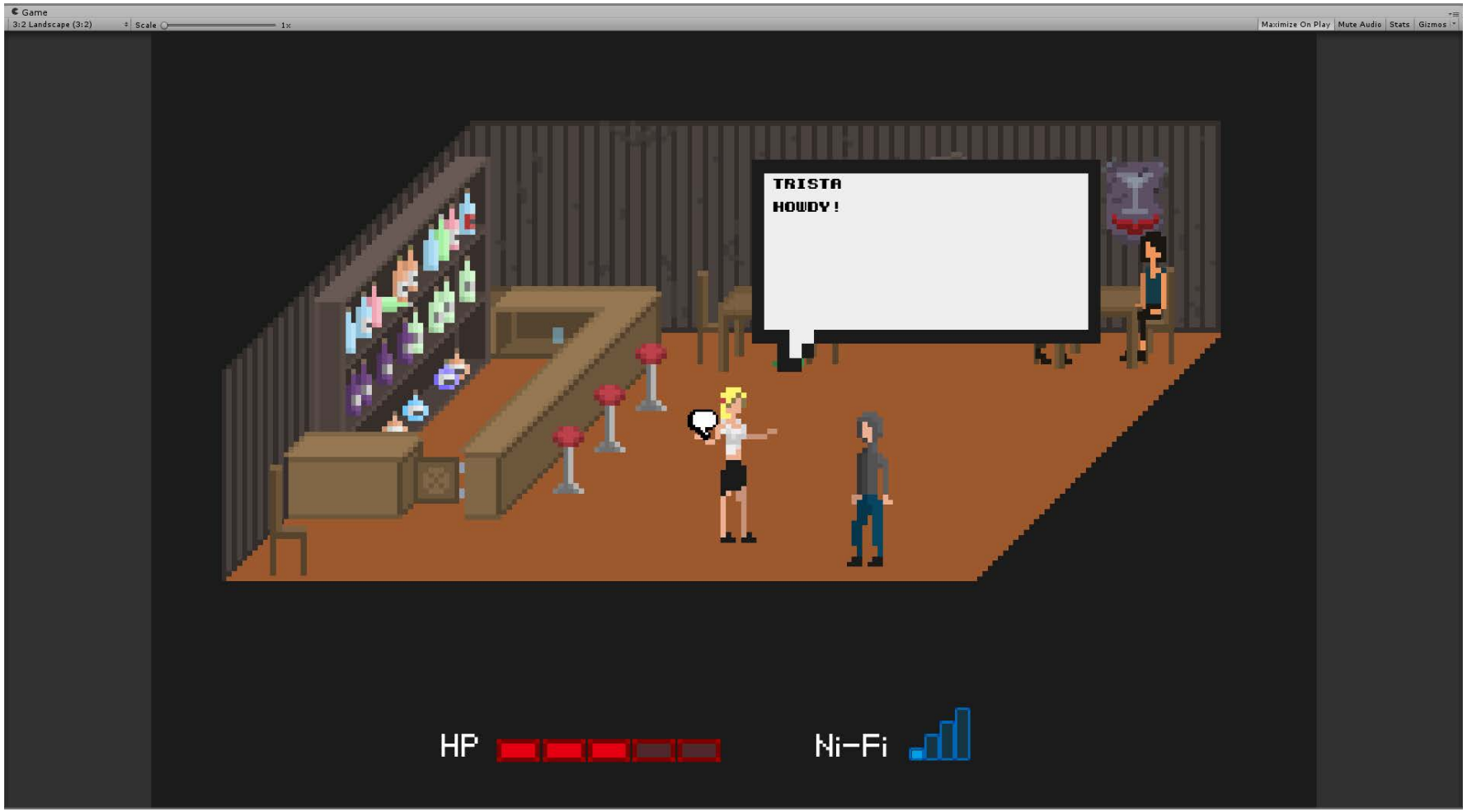
Simpleton is a collection of highly interactive short-stories you can play on your phone.

Since the stories are created by a collaboration of non-technical writers, the game needs to expose its system in a friendly, gameplay-driving interface

Simpleton

dsalgadomaia@gmail.com
http://duartemaia.com

Overview



DRIFTKING

dsalgadomaia@gmail.com
http://duartemaia.com

Dialogue Tool

- Visual Scripting under the guise of Writing -

Simpleton

Dialogue Tool

dsalgadomaia@gmail.com
http://duartemaia.com

Because the Simpleton project is a work made in collaboration with a collective of Writers (under B.C. Woodruff's supervision), I knew I had to come up with a way to enable them to drive gameplay while not being too technically demanding.

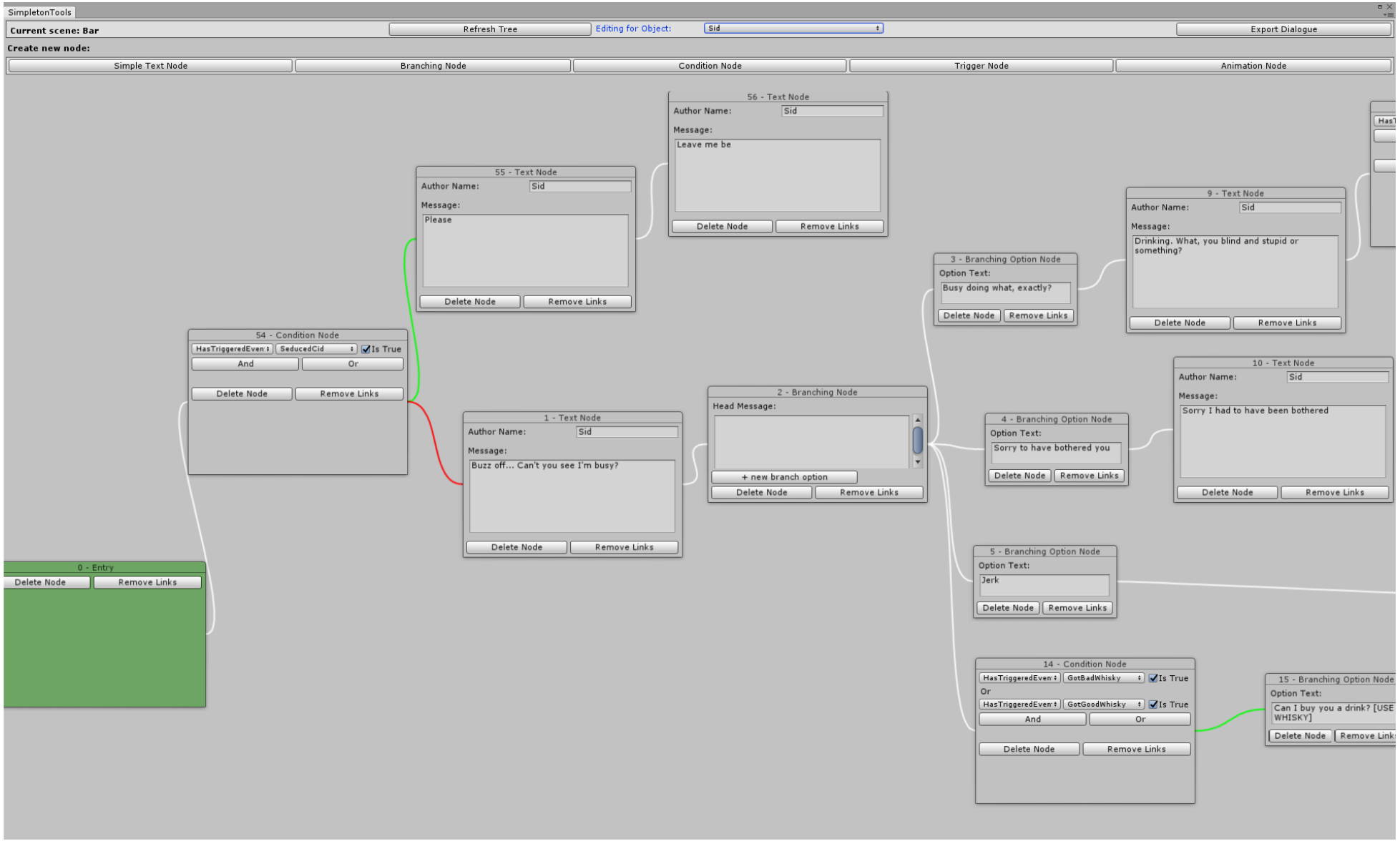
By decoupling my work from the Writer's, I would allow them to try out ideas without dependencies, and I myself would be able to work on other systems while gameplay is still being developed

The Dialogue Tool is a node-based visual scripting tool that drives dialogue, branching narratives, in-game events, npc goals & relationships, animation, cutscenes, and player state

Simpleton

Dialogue Tool

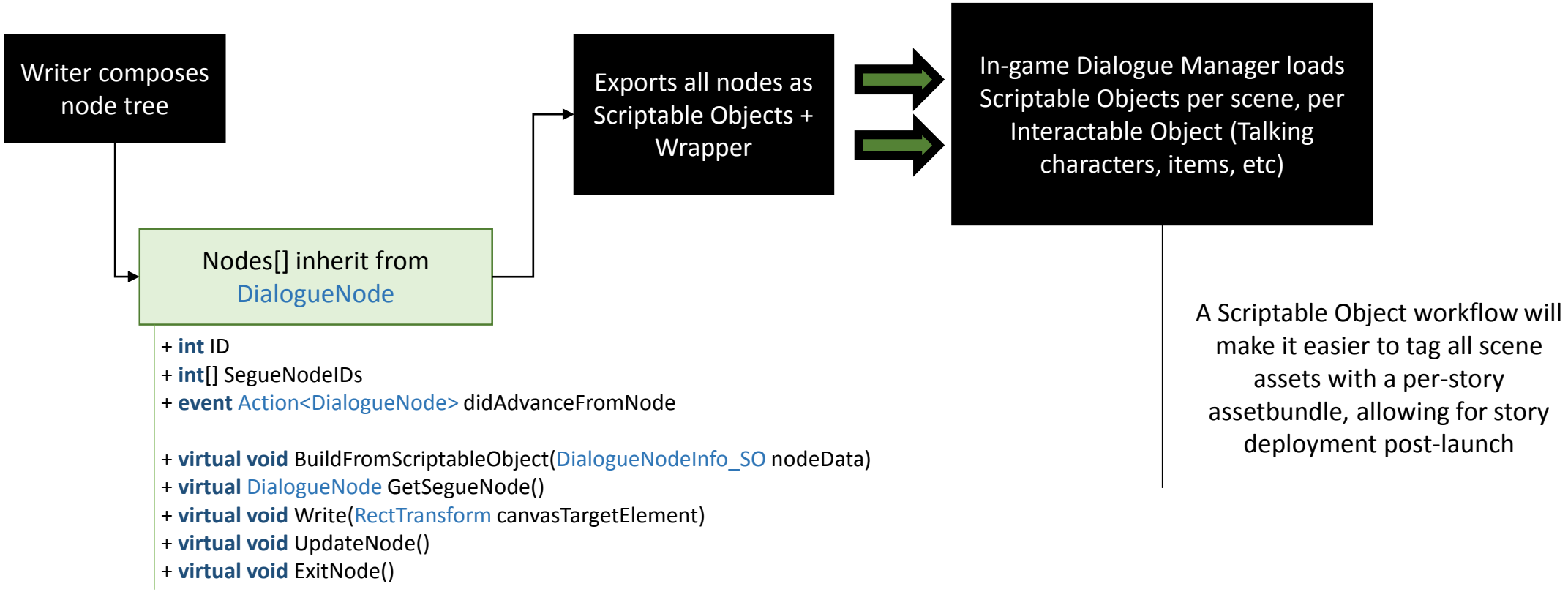
dsalgadomaia@gmail.com
http://duartemaia.com



Simpleton

Dialogue Tool

dsalgadomaia@gmail.com
http://duartemaia.com



A Scriptable Object workflow will make it easier to tag all scene assets with a per-story assetbundle, allowing for story deployment post-launch

Simpleton

Dialogue Tool

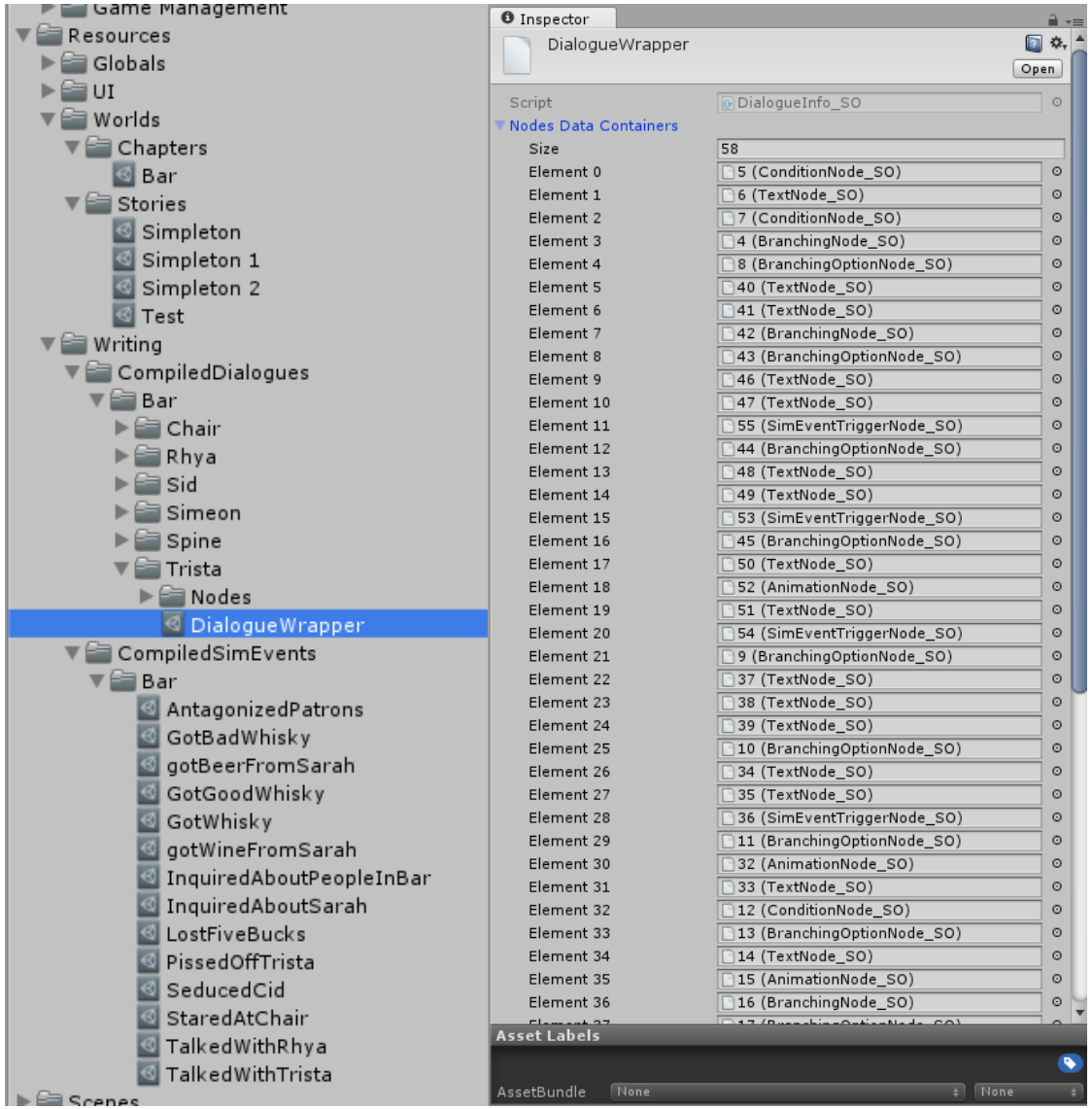
dsalgadomaia@gmail.com
http://duartemaia.com

Writers can also create and query for events they create in the tool.

The writer has then full freedom to develop his own conditional flow and branching paths, without the need for any code. These conditionals can query the inventory system, or even check the current relationship between the player and any object that can be interacted with (e.g, “animosity” < 0.3)

SimEvents are also exported as ScriptableObjects, inheriting from a few options:

- HasTriggeredEvent (catch-all)
- HasItem
- HasRelationWithInteractable(string parameter, float value)



Simpleton

Dialogue Tool Nodes

dsalgadomaia@gmail.com
http://duartemaia.com

The simplest node simply declares how to build from a ScriptableObject, and what happens when the dialogue reaches a node of its type (Write())

```
namespace Simpleton
{
  [System.Serializable]
  public class TextNode : DialogueNode
  {
    private string AuthorName;
    private string Message;
    private TextNode_SO.SpeechAlign Alignment;

    public TextNode()
    { }

    public TextNode(TextNode_SO textNodeInfoAsset)
    {
      BuildFromScriptableObject(textNodeInfoAsset);
    }

…..
```

```
  public override void Write(RectTransform intoFrame)
  {
    string speech = "";

    if (!string.IsNullOrEmpty(AuthorName))
    {
      speech += AuthorName;
      speech += "\n\n";
    }

    speech += Message;

    DialogueUIHandler.DrawSpeechText(DialogueManager.Instance.currentDialogue.Owner, speech, null);
  }

  public override void BuildFromScriptableObject(DialogueNodeInfo_SO nodeData)
  {
    base.BuildFromScriptableObject(nodeData);

    TextNode_SO textNodeInfoAsset = (TextNode_SO) nodeData;

    AuthorName = textNodeInfoAsset.AuthorName;
    Message = textNodeInfoAsset.Message;
    Alignment = textNodeInfoAsset.Alignment;
  }
 }
}
```

Simpleton

dsalgadomaia@gmail.com
http://duartemaia.com

Dialogue Tool Nodes

```
namespace Simpleton
{
  [System.Serializable]
  public class SimEventTriggerNode : DialogueNode
  {
    public SimEvent simEvent;
    SimEventTriggerNode_SO nodeDataAsset;

    public static event Action<SimEventTriggerNode> onNodeEventTrigger;

    public SimEventTriggerNode() {   }

    public SimEventTriggerNode(DialogueNodeInfo_SO nodeInfoAsset)
    {
      BuildFromScriptableObject(nodeInfoAsset);
    }

    public override void UpdateNode() { }

    public override void BuildFromScriptableObject(DialogueNodeInfo_SO nodeData)
    {
      base.BuildFromScriptableObject(nodeData);

      SimEventTriggerNode_SO data = (SimEventTriggerNode_SO)nodeData;
      nodeDataAsset = data;

      simEvent = data.simEvent;
    }

……..

    public override void Write(RectTransform intoFrame)
    {
      TriggerSimEvent();
    }


    public void TriggerSimEvent()
    {
      simEvent.hasTriggered = true;

      if (DialogueNodeSegueIDs.Count != 0)
      {
        DialogueNode nextNode = DialogueManager.Instance.currentDialogue.DialogueNodes
                                .Find((n) => n.Id == DialogueNodeSegueIDs[0]);

        FireDidAdvanceFromNodeEvent(nextNode);
      }
      else
        FireDidAdvanceFromNodeEvent(null);


      if (onNodeEventTrigger != null)
        onNodeEventTrigger(this);
    }
  }
}
```

Simpleton

dsalgadomaia@gmail.com
http://duartemaia.com

World Generator Tool I/O

```
namespace SimpletonTools
{
  public static class WG_IO
  {
    public static event Action didExportNewAsset;

    public static Story[] ImportStories() {
      return Import<Story>();
    }

    public static Chapter[] ImportChapters() {
      return Import<Chapter>();
    }

    public static void ExportStory(Story story) {
      Export<Story>(story, story.Title);
    }

    public static void ExportChapter(Chapter chapter) {
      Export<Chapter>(chapter, chapter.Title);
    }

    private static T[] Import<T>() where T:ScriptableObject  {
      string path = GetPathForWorldManagementObjectsOfType<T>(true);

      T[] assets = Resources.LoadAll<T>(path);

      if (assets == null && assets.Length < 1)
        return null;
      else
        return assets;
    }

    private static bool Export<T>(T SO, string filename = "New WorldManagement SO") where T:ScriptableObject
    {
      string path = GetPathForWorldManagementObjectsOfType<T>();

      ScriptableObject asset = ScriptableObjectUtility.CreateAsset(SO, path, filename + ".asset");

      if (didExportNewAsset != null)
        didExportNewAsset();

      return (asset != null);
    }

    private static string GetPathForWorldManagementObjectsOfType<T>(bool omitResourcesFolder = false) where T : ScriptableObject
    {
      string path;

      if (typeof(T) == typeof(Story))
        path = StoriesPathDeclarator.pathToStoriesFolder;

      else if (typeof(T) == typeof(Chapter))
        path = StoriesPathDeclarator.pathToChaptersFolder;
      else
        throw new UnityException("World Generator could not export the given type: " + typeof(T).Name);

      if (omitResourcesFolder)
        path = SimPath.GetPathRelativeToResources(path);

      return path;
    }
  }
}
```

Wrap-up!

dsalgadomaia@gmail.com
http://duartemaia.com

Thank you for your attention.

That was a snapshot of my latest work but there is much, much more that was omitted here, for brevity

Feel free to contact me if you want to know more:

dsalgadomaia@gmail.com
Phone: (604) 704-8465

You can find an overview of my latest works here:

http://duartemaia.com

And some select code snippets here

https://github.com/D-Maia