Casanova 2.0

A checkpoint-based RTS

Dr. Giuseppe Maggiore Dino Dini

NHTV University of Applied Sciences
Breda, Netherlands

Agenda

Introduction

Our goal

- Build an RTS
- Graphics in Unity (BTW: programmer's art)
- Core mechanics (no bonuses, upgrades, etc.)

Introduction

RTS

- A graph of nodes
- Nodes grouped into areas
- Only links can be traversed by units
- Only enable/disable automated sending along a link
- No direct control over units, just “open/close the faucet”

World

```
worldEntity World =
  {
    CurrentSelection        : CurrentSelection
    StarSystems             : [StarSystem]
    Links                   : [Link]
    Players                 : [Player]
```

World

```
Create(initialPlayers : [string*Color],unityMap:
    UnityMap) =
  let players =
    [ for name, color in initialPlayers do
      select (Player.Create(name, color)) ]
```

World

```
let star_systems =
  [ for star_system in unityMap.StarSystems do
    select (StarSystem.Create(current_selection,
        star_system, players)) ]
```

World

```
let all_checkpoints =
  [ for star_system in star_systems do
    selectMany star_system.Checkpoints ]
```

World

```
let links =
  [ for link in unityMap.Links do
    let beginning =
      [ for checkpoint in all_checkpoints do
        findBy (checkpoint.UnityCheckpoint.
            gameObject = link.
            BeginningGameObject) ]
    let ending =
      [ for checkpoint in all_checkpoints do
        findBy (checkpoint.UnityCheckpoint.
            gameObject = link.EndGameObject) ]
    select (Link.Create(link, beginning, ending)
        ) ]
```

World

```
{
  CurrentSelection        = CurrentSelection.
      Create()
  StarSystems             = star_systems
  Links                   = links
  Players                 = players
}
```

Selection

```
entity CurrentSelection =
  {
    SelectionRectangle          : Option<
        UnitySelectionRectangle>
```

Selection

```
Create() =
  {
    SelectionRectangle     = None
    SelectedDestination    = None
    SelectedSources        = []
  }
```

Selection

```
rule SelectionRectangle =
  when SelectionRectangle = None && Input.
      GetMouseButton(0)
  yield Some(UnitySelectionRectangle.Create())
```

Selection

```
rule SelectionRectangle.Value.Destroyed,
    SelectionRectangle.Value.EnableSelection =
  when SelectionRectangle <> None && not(Input.
      GetMouseButton(0))
  yield false,true
  yield true,false
```

Selection

```
rule SelectionRectangle =
  when SelectionRectangle <> None &&
       SelectionRectangle.Value.Destroyed
  yield None
```

Player

```
entity Player =
  {
    Name              : string
    Color             : Color
    UnityPlayer       : UnityPlayer
```

Player

```
Create(name, color, camera:UnityCamera) =
  {
    Name                = name
    Color               = color
    UnityPlayer         = UnityPlayer.Create(name,
        color)
  }
```

Player

```
rule UnityPlayer.NumArmies =
  yield
    [ for star_system in world.StarSystems do
      for checkpoint in star_system.Checkpoints do
      where (checkpoint.Owner = this)
      sumBy (checkpoint.LocalFleet.NumShips) ]
  wait 1.0f<s>
```

StarSystem

```
entity StarSystem =
  {
    Checkpoints                    : [Checkpoint]
```

StarSystem

```
    Create(star_system:UnityStarSystem) =
      {
        Checkpoints =
          [ for checkpoint in star_system.Checkpoints
              do
            select (Checkpoint.Create(checkpoint)) ]
      }
  }
```

Checkpoint

```
entity Checkpoint =
  {
    UnityCheckpoint              : UnityCheckpoint
    ref Owner                    : Option<Player>
    LocalFleet                   : Fleet
    Attackers                    : [AttackingFleet]
```

Checkpoint

```
Create(unity_checkpoint) =
  {
    UnityCheckpoint          = unity_checkpoint
    Owner                    = unity_checkpoint.
        InitialOwner
    LocalFleet               = Fleet.Create( 0)
    Attackers                = []
  }
```

Checkpoint

```
rule UnityCheckpoint.OwnerColor =
  when Owner <> None
  yield Owner.Value.Color

rule UnityCheckpoint.NumShips =
  when Ownership.Owner <> None
  yield LocalFleet.NumShips.ToString()
```

Checkpoint

```
rule UnityCheckpoint.IsTarget =
  when UnityCheckpoint.IsTarget
  yield true
  wait 0.1f<s>
  yield false
```

Checkpoint

```
rule UnityCheckpoint.Attackers =
  when Attackers <> []
  yield
    [ for attacker in Attackers do
      groupBy attacker.Owner into attackGroup
      let owner = attack_group.Key
      let numShips =
        [ for fleet in attackGroup.Elements do
          sumBy attackGroup.NumShips ]
      select UnityText.Create(numShips, owner.
          Color) ]
```

Checkpoint

```
rule LocalFleet.NumShips, Attackers =
  when Attackers <> []
  let num_attackers = Attackers.Length
  wait 0.1f<s>
  yield LocalFleet.NumShips - num_attackers,
        [ for a in Attackers do
          select { a with Fleet.NumShips = a.Fleet
              .NumShips - 1 } ]
```

Checkpoint

```
rule Attackers =
  [ for a in Attackers do
    where (a.Fleet.NumShips > 0)
    select a ]
```

Checkpoint

```
rule LocalFleet, Owner, Attackers =
  when LocalFleet.NumShips <= 0 && Attackers <> []
  let newOwnersAttackers =
        [ for a in Attackers do
          where (a.Owner = Attackers.Head.Owner)
          select a ]
  let newLocalFleet =
        [ for a in newOwnersAttackers do
          sumBy a.Fleet.NumShips ]
  yield Fleet.Create(newLocalFleet),
        Some Attackers.Head.Owner,
        Attackers - newOwnersAttackers
```

Link

```
entity Link =
  {
    ref UnityLink          : UnityLink
    ref Start              : Checkpoint
    ref End                : Checkpoint
    Fleets                 : [TravelingFleet]
    ArrivedFleets          : [TravelingFleet]
    AutoSendEnabled        : bool
```

Link

```
Create(link:UnityLink,s,e,link_bonuses) =
  {
    UnityLink         = link
    Start             = s
    End               = e
    Fleets            = []
    ArrivedFleets     = []
    AutoSendEnabled = false
  }
```

Link

```
rule AutoSendEnabled =
  when Start.UnityCheckpoint.IsSource && End.
      UnityCheckpoint.IsTarget
  yield not AutoSendEnabled
```

Link

```
rule Fleets,Start.LocalFleet.NumShips =
  when AutoSendEnabled && Start.LocalFleet.
      NumShips >= 0 &&
        Start.Owner <> None
  let sent_fleets = Start.LocalFleet.NumShips / 5
  let new_fleet =
   TravelingFleet.Create(
     Start.UnityCheckpoint.Position,End,
     sent_fleets,Start.Owner.Value)
  yield new_fleet :: Fleets, Start.LocalFleet.
      NumShips - sent_fleets
  wait 1.0f<s>
```

Link

```
rule ArrivedFleets =
  [ for f in Fleets do
    where f.UnityShip.Destroyed
    select f ]
```

Link

```
rule Fleets = Fleets - ArrivedFleets
```

Link

```
rule End.LocalFleet.NumShips =
  let arrived_fleets =
    [ for f in ArrivedFleets do
      where (Some f.Owner = End.Ownership.Owner)
      select f.Fleet.NumShips
      sum ]
  End.LocalFleet.NumShips + arrived_fleets
```

Link

```
rule End.Attackers =
  End.Attackers @
    [ for f in ArrivedFleets do
      where (Some f.Owner <> End.Owner)
      select (AttackingFleet.Create(f.Fleet.
          NumShips,f.Owner)) ]
```

Fleet

```
entity Fleet =
  {
    NumShips     : int
    Create(num_ships) =
      {
        NumShips = num_ships
      }
  }
```

Fleet

```
entity AttackingFleet =
  {
    Fleet      : Fleet
    Owner      : Player

    Create(num_ships,owner) =
      {
        Fleet      = Fleet.Create(num_ships)
        Owner      = owner
      }
  }
```

Fleet

```
entity TravelingFleet =
  {
    UnityShip           : UnityShip
    Fleet               : Fleet
    Owner               : Player
    ref Target          : Checkpoint
    ref LinkBonuses     : LinkBonuses
```

Fleet

```
    Create(position,target:Checkpoint,num_ships,owner)
        =
      let Unity_ship = UnityShip.Create(position,owner
          .Color)
      {
        UnityShip      = Unity_ship
        Fleet          = Fleet.Create(num_ships)
        Owner          = owner
        Target         = target
      }
  }
```

Fleet

```
rule UnityShip.Velocity =
  Vector3.Normalize(Target.UnityCheckpoint.
      Position - UnityShip.Position)

rule UnityShip.Destroyed =
  Vector3.Distance(UnityShip.Position, Target.
      UnityCheckpoint.Position) <= Target.
      UnityCheckpoint.Radius + UnityShip.Radius
```

Pathfinding

```
entity PathContext =
  {
    Distance     : float32
    ref Steps    : [Link]
    Create(dist,steps) =
      {
        Distance = dist
        Steps = steps
      }
  }
```

Pathfinding

```
entity Pathfinding =
  {
    ref MinimumPaths : [Checkpoint -> [Checkpoint ->
        PathContext]]
```

Pathfinding

```
Create(all_checkpoints : [Checkpoint],links : [
    DirectedLink]) =
```

Pathfinding

```
let neighbors =
  [ for c in all_checkpoints do
    select(c,[ for l in links do
               where (l.Start = c && l.End <> c)
               select l ])
    toMap ]
```

Pathfinding

```
let minimumPaths =
  [ for checkpoint in all_checkpoints do
    select (checkpoint,
            [ for (k,c) in [checkpoint,
                PathContext.Create(0.0f,[])] do
              closeOver [ for neighbor in
                neighbors.[k] do
                  let dist' = c.Distance
                    + neighbor.
                    UnityLink.Length
                  let steps' = neighbor
                    :: c.Steps
                  select (neighbor.End,
                    PathContext.Create
                    (dist', steps')) ]
```

Pathfinding

```
    {
        MinimumPaths = minimumPaths
    }
```

Conclusion

What did we just see?

- A simple, yet full, RTS
- Graphics in Unity; graphics is a bit of a solved problem
- Core mechanics (no bonuses, upgrades, etc.)
- Advanced mechanics are not that much more (pathfinding)

Conclusion

Why?

- Stress-test of Casanova
- Stress-test of integration with Unity

Conclusion

Why?

- Stress-test of Casanova: works perfectly
- Stress-test of integration with Unity: works well, but still improving

Conclusion

Things that work

- Most state machines/queries would turn into a nightmare
- Virtually no debugging happened during development (yup, it is not a joke)
- (Extended version of the game took exactly 7 days to make)
- System works, we even have a starter kit example

Conclusion

Things that could work better

- Optimization of queries and events
- Automated binding of entities and scene
- Automated compilation of Casanova files that change

Conclusion

A look ahead

- Very aggressive forms of optimization
- Networking primitives
- More testing and sample games

That's it

Thank you!