Buttery-smooth **Scrolling** with **Texture/AsyncDisplayKit**

Taufik Obet, iOS Developer at Happy5

A brief history

— An open-source project by Facebook and Pinterest

— Debut on Facebook Paper app

Goal

— Maintain steady 60 fps scrolling performance with complex layout by default.

Real-world Apps (I personally tried)

— ~~Facebook Paper~~ (pulled down from App Store)
— Pinterest
— Buffer

ASDKGram Demo

Twitch Streams API

— Quick JSON parsing with Swift 4 `Codable` protocol

GET https://api.twitch.tv/kraken/streams/


```
{
   "streams": [
   {
      "_id": 23937446096,
      "channel": {
            "_id": 121059319,
            "status": "Playing Overwatch with friends",
            "url": "https://www.twitch.tv/moonmoon_ow",
            "display_name": "MOONMOON_OW",
            "logo": "https://static-cdn.jtvnw.net/useravatar.png"

      },
      "preview": {
            "large": "https://static-cdn.jtvnw.net/large-preview.jpg"
      },
      "video_height": 720,
      "viewers": 11523,
      "game": "Overwatch"
   },
      ...
   ]
}
```

```
struct TwitchStreamsService: Codable {
    let streams:[TwitchStream]

}

struct TwitchStream: Codable {
    let id: Int
    let channel:TwitchChannel
    let preview:VideoPreview
    let videoHeight:Int
    let viewers:Int
    let game:String

    enum CodingKeys: String, CodingKey {
        case id = "_id"
        case channel
        case preview
        case videoHeight = "video_height"
        case viewers
        case game
    }
}
```

```
struct TwitchChannel: Codable {
    let id: Int
    let status: String
    let url: URL
    let displayName: String
    let avatar:URL

    enum CodingKeys: String, CodingKey {
        case id = "_id"
        case status
        case url
        case displayName = "display_name"
        case avatar = "logo"
    }
}

struct VideoPreview: Codable {
    let largeURL: URL

    enum CodingKeys: String, CodingKey {
        case largeURL = "large"
    }
}
```

ASDisplayNode

— `UIView`-like API

— `addSubnode` instead of `addSubview`

— Wraps `UIView` and `CALayer` (exposed as `.view_`, and `_.layer`, respectively)

— Thread-safe (can be instantiated on background thread)

Node Containers

— Subnode manager, manages intelligent preloading of its nodes

— `ASViewController` instead of `UIViewController`

— `ASTableNode` instead of `UITableView`

— `ASCollectionNode` instead of `UICollectionView`

ASNetworkImageNode

— Load URL asynchronously

```
let imageNode = ASNetworkImageNode()
imageNode.url = URL(string: "https://imagecdn.com/photo.jpg")
```

— Optional image processing on background thread

```
imageNode.imageModificationBlock = { image in
    return image.makeCircularImage(size: imageSize)
}
```

ASLayoutSpec

— Declarative layout

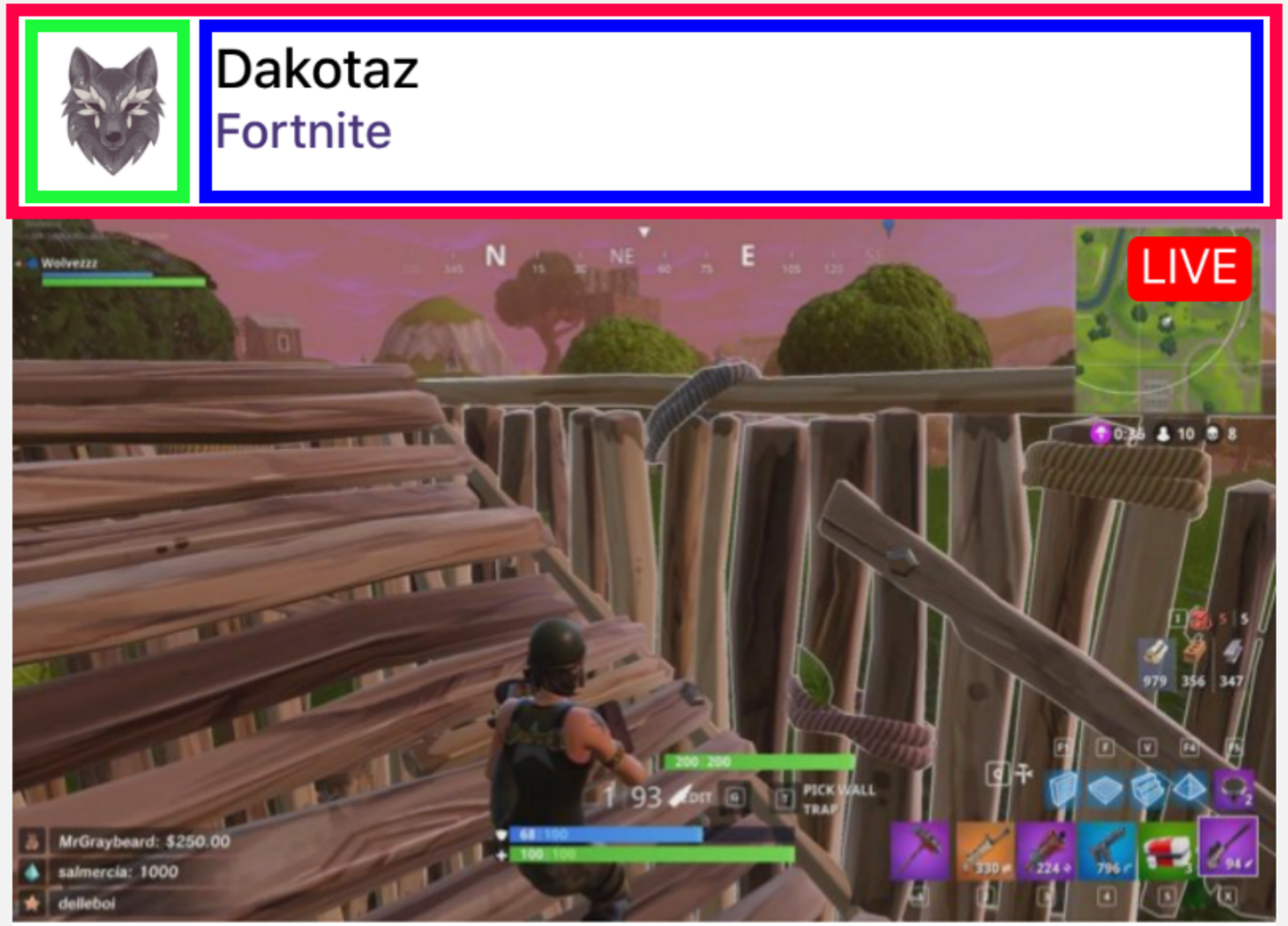
— Borrow CSS Flexbox concept

— Accept any child object with conformance to `ASLayoutElement` protocol
(e.g. `ASDisplayNode` & `ASLayoutSpec` and its subclass)

```
override func layoutSpecThatFits(_ constrainedSize: ASSizeRange) -> ASLayoutSpec {
    let displayNameStack = ASStackLayoutSpec(
                            direction: .vertical,
                            spacing: 0,
                            justifyContent: .start,
                            alignItems: .start,
                            children: [displayNameNode, gameLabelNode])

    let headerStack = ASStackLayoutSpec(
                            direction: .horizontal,
                            spacing: 8,
                            justifyContent: .start,
                            alignItems: .start,
                            children: [avatarNode, displayNameStack])

    return headerStack
}
```



headerStack

displayNameStack

avatarNode

TwitchStream app demo (open-source)

https://github.com/taufikobet/TwitchStream

More Resources

— NSLondon 2014 (Scott Goodson talk)
https://www.youtube.com/watch?v=h4QDbgB7RLo

— Texture official website and documentations
http://texturegroup.org

Thank you!

We’re Hiring!

— Swift or Objective-C programmers
— Send your resume to obet@happy5.co
— Web front-end, too! (We use React)