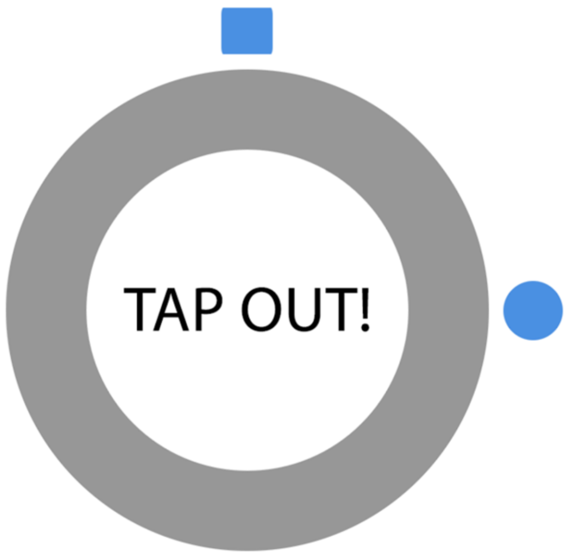
TAP OUT!

A research based project

for

SARDAR BHAGWANT SINGH PROJECT



A Novel Approach To Treating DCD
(Developmental Coordination Disorder)

Table Of Contents

Question/Proposal

Has it ever occurred to you that in some situations that our mind can't coordinate between our left and right side functions because of mental confusion?
This problem is coined as DCD (Developmental Coordination Disorder)

While this may happen to some of us in a tricky situation, this problem persists in some children by birth. DCD is a motor skills disorder that affects five to six percent of all school-aged children. It occurs when a delay in the development of motor skills, or difficulty coordinating movements, results in a child being unable to perform common, everyday tasks. By definition, children with DCD do not have an identifiable medical or neurological condition that explains their coordination problems. DCD is a major issue not only from the medical point of view but also physiologically.
Kids suffering from this disorder are often described as "clumsy" or "awkward" by their parents and teacher because children with DCD have difficulty mastering simple motor activities, such as tying shoes or going down stairs, and are unable to perform age-appropriate academic and self-care tasks. Some children may experience difficulties in a variety of areas while others may have problems only with specific activities.

Since the first description of a syndrome of clumsiness as a developmental disorder in the 1930s, and refinement of the term to "clumsy child syndrome" in 1975, children with motor coordination problems have been given a variety of labels over the past few decades including, but not limited to: developmental dyspraxia, minimal brain dysfunction, perceptual-motor dysfunction, physically awkward, and specific developmental disorder of motor function. The disorder did not gain legitimacy as a health problem until 1994, when an international panel of experts was convened at a consensus meeting held in London, Ontario. At that meeting, a decision was made to recognize "clumsy" children as having Developmental Coordination Disorder (DCD). DCD is the term that was introduced in the Diagnostic and Statistical Manual in 1989 and that has been retained in more recent editions of the manual.
This brings us to the conclusion that DCD is a major speed breaker for the kids suffering from it. This disease is not exactly curable but we, Shreyank Juneja and Rohan Parmar as a part of a brilliant billion piece blueprint called the earth, have tried our best to help these kids to help our species suffering from this mammoth of a problem by developing an app that puts our coordination to test.

Research

Developmental coordination disorder (DCD) also known as developmental dyspraxia or quite simply dyspraxia, is a chronic neurological disorder beginning in childhood that can affect planning of movements and co-ordination as a result of brain messages not being accurately transmitted to the body. Developmental coordination disorder is diagnosed in the absence of other neurological impairments like cerebral palsy, muscular dystrophy, multiple sclerosis or Parkinson's disease. It affects 5 to 6 percent of school-aged children.

Signs and symptoms:

Various areas of development can be affected by developmental coordination disorder and these will persist into adulthood,[10] as DCD has no cure. Often various coping strategies are developed, and these can be enhanced through occupational therapy, psychomotor therapy, physiotherapy, speech therapy, or psychological training.
In addition to the physical impairments, developmental coordination disorder is associated with problems with memory, especially working memory. This typically results in difficulty remembering instructions, difficulty organizing one's time and remembering deadlines, increased propensity to lose things or problems carrying out tasks which require remembering several steps in sequence (such as cooking). Whilst most of the general population experience these problems to some extent, they have a much more significant impact on the lives of dyspraxic people.However, many dyspraxics have excellent long-term memories, despite poor short-term memory. Many dyspraxics benefit from working in a structured environment, as repeating the same routine minimises difficulty with time-management and allows them to commit procedures to long-term memory.
People with developmental coordination disorder sometimes have difficulty moderating the amount of sensory information that their body is constantly sending them, so as a result dyspraxics are prone to sensory overload and panic attacks.
Many dyspraxics struggle to distinguish left from right, even as adults, and have extremely poor sense of direction generally.
Moderate to extreme difficulty doing physical tasks is experienced by some dyspraxics, and fatigue is common because so much extra energy is expended while trying to execute physical movements correctly. Some (but not all) dyspraxics suffer from hypertonia, low muscle tone, which like DCD can detrimentally affect balance.

Gross motor control

Whole body movement, motor coordination, and body image issues mean that major developmental targets including walking, running, climbing and jumping

can be affected. The difficulties vary from person to person and can include the following:
• Poor timing
• Poor balance (sometimes even falling over in mid-step). Tripping over one's own feet is also common.
• Difficulty combining movements into a controlled sequence.
• Difficulty remembering the next movement in a sequence.
• Problems with spatial awareness, or proprioception.
• Some people with developmental coordination disorder have trouble picking up and holding onto simple objects such as pencils, owing to poor muscle tone and/or proprioception.
• This disorder can cause an individual to be clumsy to the point of knocking things over and bumping into people accidentally.
• Some people with developmental coordination disorder have difficulty in determining left from right.
• Cross-laterality, ambidexterity, and a shift in the preferred hand are also common in people with developmental coordination disorder.
• Problems with chewing foods.

Fine motor control

Fine-motor problems can cause difficulty with a wide variety of other tasks such as using a knife and fork, fastening buttons and shoelaces, cooking, brushing one's teeth, styling one's hair, shaving, applying cosmetics, opening jars and packets, locking and unlocking doors, and doing housework.
Difficulties with fine motor co-ordination lead to problems with handwriting, which may be due to either ideational or ideo-motor difficulties. Problems associated with this area may include:
• Learning basic movement patterns.
• Developing a desired writing speed.
• Establishing the correct pencil grip
• The acquisition of graphemes – e.g. the letters of the Latin alphabet, as well as numbers.

Developmental verbal dyspraxia

Developmental verbal dyspraxia (DVD) is a type of ideational dyspraxia, causing speech and language impairments. This is the favoured term in the UK; however, it is also sometimes referred to as articulatory dyspraxia, and in the United States the usual term is childhood apraxia of speech (CAS).
Key problems include:
• Difficulties controlling the speech organs.
• Difficulties making speech sounds
• Difficulty sequencing sounds
• Within a word
• Forming words into sentences
• Difficulty controlling breathing, suppressing salivation and phonation when talking or singing with lyrics.
• Slow language development

Method/Testing and Design

The design is an interesting and efficient ensemble of beautifully executed Swift code. The ensemble consists of various Swift files that are imported at various instances of code.
The files are as follows:

1. `AppDelegate.Swift:`

```
//
//  AppDelegate.swift
//  ColorTap
//
//
//  Created by Rohan Parmar on 28/7/16.
//  Copyright (c) 2016 Red Circle Studios. All rights reserved.
//

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive
state. This can occur for certain types of temporary interruptions (such as an
incoming phone call or SMS message) or when the user quits the application and
it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and throttle
down OpenGL ES frame rates. Games should use this method to pause the game.
```

```
            }

            func applicationDidEnterBackground(application: UIApplication) {
                // Use this method to release shared resources, save user data,
invalidate timers, and store enough application state information to restore your
application to its current state in case it is terminated later.
                // If your application supports background execution, this method is
called instead of applicationWillTerminate: when the user quits.
            }

            func applicationWillEnterForeground(application: UIApplication) {
                // Called as part of the transition from the background to the inactive
state; here you can undo many of the changes made on entering the background.
            }

            func applicationDidBecomeActive(application: UIApplication) {
                // Restart any tasks that were paused (or not yet started) while the
application was inactive. If the application was previously in the background,
optionally refresh the user interface.
            }

            func applicationWillTerminate(application: UIApplication) {
                // Called when the application is about to terminate. Save data if
appropriate. See also applicationDidEnterBackground:.
            }


        }
```

2. GameScene.Swift

```
//
//  GameScene.swift
//  ColorTap
//
//  Created by Rohan Parmar on 28/7/16.
//  Copyright (c) 2016 Red Circle Studios. All rights reserved.
//

import SpriteKit

class GameScene: SKScene {

    var Circle = SKSpriteNode()
    var Person = SKSpriteNode()
    var Dot = SKSpriteNode()
    var Path = UIBezierPath()
    var LevelLabel = UILabel()
    var currentLevel = Int()
    var currentScore = Int()
    var highLevel = Int()
    var tries = Int()
    var lives = UILabel()

    var gameStarted = Bool()
    var movingClockWise = Bool()
    var intersected = false
    override func didMoveToView(view: SKView) {
        /* Setup your scene here */
        loadView()
        let defaults = NSUserDefaults.standardUserDefaults()
        if defaults.integerForKey("HighLevel") != 0 {
            highLevel = defaults.integerForKey("HighLevel") as Int!
            currentLevel = highLevel
            currentScore = currentLevel
            LevelLabel.text = "\(currentScore)"
        } else {
            defaults.setInteger(1, forKey: "HighLevel")
        }
    }

    func loadView() {
        self.view?.backgroundColor = UIColor.whiteColor()
        movingClockWise = true
        Circle = SKSpriteNode(imageNamed: "Circle")
        Circle.size = CGSize(width: 170, height: 170)
        Circle.position = CGPoint(x: CGRectGetMidX(self.frame), y:
CGRectGetMidY(self.frame))
        self.addChild(Circle)
        Person = SKSpriteNode(imageNamed: "Person")
        Person.size = CGSize(width: 20, height: 20)
        Person.position = CGPoint(x: self.frame.width / 2, y: self.frame.height / 2 + 112)
        Person.zRotation = 3.14 / 2
        Person.zPosition = Circle.zPosition + 2
        self.addChild(Person)
        addDot()
        LevelLabel = UILabel(frame: CGRect(x: 0, y: 0, width: 150, height: 100))
        LevelLabel.center = (self.view?.center)!
        LevelLabel.text = "\(currentScore)"
        LevelLabel.textAlignment = NSTextAlignment.Center
        LevelLabel.font = UIFont.systemFontOfSize(60)
        LevelLabel.textColor = SKColor.darkGrayColor()
        self.view?.addSubview(LevelLabel)
        lives = UILabel(frame: CGRect(x: 0, y: 0, width: 150, height: 50))
        lives.center = CGPoint(x: 187, y: 200)
        lives.text = "Tries: \(tries)"
        lives.textAlignment = NSTextAlignment.Center
        lives.font = UIFont.systemFontOfSize(20)
        lives.textColor = SKColor.darkGrayColor()
        self.view?.addSubview(lives)
    }

    override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
        /* Called when a touch begins */
        if gameStarted == false {
            moveClockWise()
            movingClockWise = true
            gameStarted = true
        } else if gameStarted == true {
            if movingClockWise == true {
                moveCounterClockWise()
                movingClockWise = false
            } else if movingClockWise == false {
                moveClockWise()
                movingClockWise = true
            }
            DotTouched()
        }
    }

    func moveClockWise() {
        let dx = Person.position.x - self.frame.width / 2
        let dy = Person.position.y - self.frame.height / 2
        let rad = atan2(dy, dx)
        Path = UIBezierPath(arcCenter: CGPoint(x: self.frame.width / 2, y:
self.frame.height / 2), radius: 100, startAngle: rad, endAngle: rad + CGFloat(M_PI * 4),
clockwise: true)
        let follow = SKAction.followPath(Path.CGPath, asOffset: false, orientToPath: true,
speed: 300)
        Person.runAction(SKAction.repeatActionForever(follow).reversedAction())
    }

    func moveCounterClockWise() {
        let dx = Person.position.x - self.frame.width / 2
        let dy = Person.position.y - self.frame.height / 2
        let rad = atan2(dy, dx)
        Path = UIBezierPath(arcCenter: CGPoint(x: self.frame.width / 2, y:
self.frame.height / 2), radius: 100, startAngle: rad, endAngle: rad + CGFloat(M_PI * 4),
clockwise: true)
        let follow = SKAction.followPath(Path.CGPath, asOffset: false, orientToPath: true,
speed: 300)
        Person.runAction(SKAction.repeatActionForever(follow))
    }


    func addDot() {
        Dot = SKSpriteNode(imageNamed: "Dot")
        Dot.size = CGSize(width: 20, height: 20)
        Dot.zPosition = Circle.zPosition + 2
        let dx = Person.position.x - self.frame.width / 2
        let dy = Person.position.y - self.frame.height / 2
        let rad = atan2(dy, dx)

        if movingClockWise == true {
            let tempAngle = CGFloat.random(min: rad - 1.0, max: rad - 2.5)
            let Path2 = UIBezierPath(arcCenter: CGPoint(x: self.frame.width / 2, y:
self.frame.height / 2), radius: 100, startAngle: tempAngle, endAngle: tempAngle +
CGFloat(M_PI * 4), clockwise: true)
            Dot.position = Path2.currentPoint
        } else if movingClockWise == false {
            let tempAngle = CGFloat.random(min: rad + 1.0, max: rad + 2.5)
            let Path2 = UIBezierPath(arcCenter: CGPoint(x: self.frame.width / 2, y:
self.frame.height / 2), radius: 100, startAngle: tempAngle, endAngle: tempAngle +
CGFloat(M_PI * 4), clockwise: true)
            Dot.position = Path2.currentPoint
        }
        self.addChild(Dot)
    }

    func DotTouched() {
        if intersected == true {
            Dot.removeFromParent()
            addDot()
            intersected = false
            currentScore -= 1
            LevelLabel.text = "\(currentScore)"
            lives.text = "Tries: \(tries)"
            if currentScore <= 0 {
                nextLevel()
            }
        } else if intersected == false {
            died()
        }
    }

    func nextLevel() {
        currentLevel += 1
        currentScore = currentLevel
        LevelLabel.text = "\(currentScore)"
        lives.text = "Tries: \(tries)"
        won()
        if currentLevel > highLevel {
            highLevel = currentLevel
            let defaults = NSUserDefaults.standardUserDefaults()
            defaults.setInteger(highLevel, forKey: "HighLevel")
        }
    }

    func died() {
        self.removeAllChildren()
        let action1 = SKAction.colorizeWithColor(UIColor.redColor(), colorBlendFactor:
1.0, duration: 0.2)
        let action2 = SKAction.colorizeWithColor(UIColor.whiteColor(), colorBlendFactor:
1.0, duration: 0.2)
        //self.scene?.runAction(SKAction.sequence([action1, action2])
        self.scene?.runAction(SKAction.sequence([action1, action2]))
        intersected = false
        gameStarted = false
        currentScore = currentLevel
        LevelLabel.removeFromSuperview()
        tries += 1
        lives.removeFromSuperview()
        self.loadView()


    }

    func won() {
        self.removeAllChildren()
        let action1 = SKAction.colorizeWithColor(UIColor.greenColor(), colorBlendFactor:
1.0, duration: 0.2)
        let action2 = SKAction.colorizeWithColor(UIColor.whiteColor(), colorBlendFactor:
1.0, duration: 0.2)
        //self.scene?.runAction(SKAction.sequence([action1, action2])
        self.scene?.runAction(SKAction.sequence([action1, action2]))
        intersected = false
        gameStarted = false
        LevelLabel.removeFromSuperview()
        tries = 0
        lives.removeFromSuperview()
        self.loadView()

    }
    override func update(currentTime: CFTimeInterval) {
        /* Called before each frame is rendered */
        if Person.intersectsNode(Dot) {
            intersected = true
        } else {
            if intersected == true {
                if Person.intersectsNode(Dot) == false {

                    died()
                }
            }
        }
    }
}
```

3. CGFloatExtension.Swift

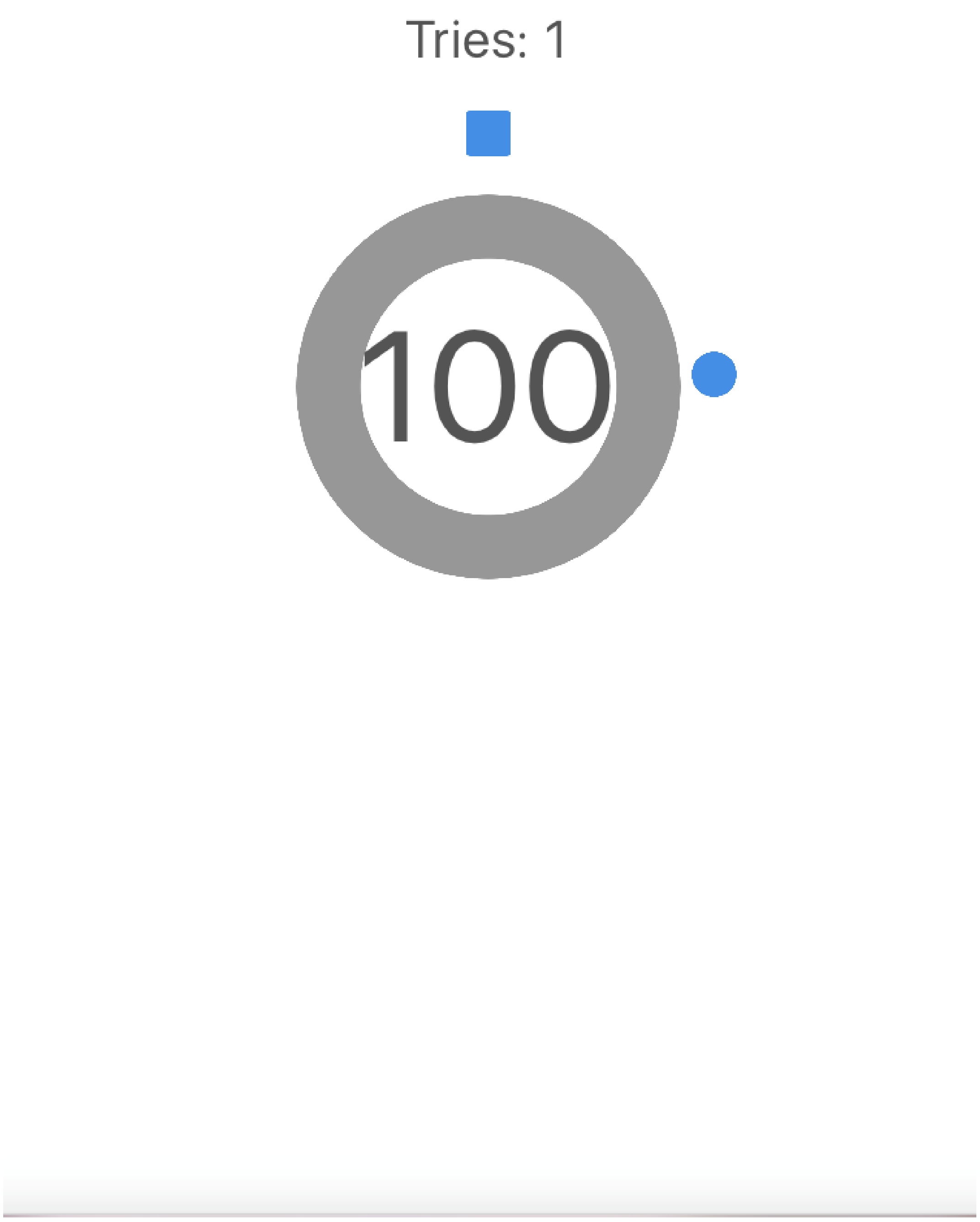
```
//
//  CGFloatExtension.swift
//  ColorTap
//
//  Created by Rohan Parmar on 28/7/16.
//  Copyright (c) 2016 Red Circle Studios. All rights reserved.
//

import Foundation
import CoreGraphics

public extension CGFloat {
    public static func random() -> CGFloat {
        return CGFloat(Float(arc4random()) / 0xFFFFFFFF)
    }
    public static func random(min min: CGFloat, max: CGFloat) -> CGFloat {
        return CGFloat.random() * (max-min) + min
    }
            }
```

4. GameViewController.Swift

```
//
//  GameViewController.swift
//  ColorTap
//
//
//  Created by Rohan Parmar on 28/7/16.
//  Copyright (c) 2016 Red Circle Studios. All rights reserved.
//
```

```
import UIKit
import SpriteKit

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let scene = GameScene(fileNamed:"GameScene") {
            // Configure the view.
            let skView = self.view as! SKView
            skView.showsFPS = true
            skView.showsNodeCount = true

            /* Sprite Kit applies additional optimizations to improve rendering
performance */
            skView.ignoresSiblingOrder = true

            /* Set the scale mode to scale to fit the window */
            scene.scaleMode = .AspectFill

            skView.presentScene(scene)
        }
    }

    override func shouldAutorotate() -> Bool {
        return true
    }

    override func supportedInterfaceOrientations() ->
UIInterfaceOrientationMask {
        if UIDevice.currentDevice().userInterfaceIdiom == .Phone {
            return .AllButUpsideDown
        } else {
            return .All
        }
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Release any cached data, images, etc that aren't in use.
    }

    override func prefersStatusBarHidden() -> Bool {
        return true
    }
}
```

The rendered result looks like this(after playing a lot for debugging reasons):


Tries: 1
100

Testing

Over the course of the next few months (Post September), we plan to work with 10 test subjects and subject them to conditional training. After the end of a one week term, we plan to record changes, inmprovements etc. in things like Stability, Hand-Eye Coordination and reaction time with respect to various challenges.
During this term, we plan to subject the test subjects to continual usage of the application we have developed and try to curb the effects of DCD.
As far as Statistical Significance is concerned, here is the algorithm we would follow:

Whenever we perform a significance test, it involves comparing a test value that we have calculated to some critical value for the statistic. It doesn't matter what type of statistic we are calculating (e.g., a t-statistic, a chi-square statistic, an F-statistic, etc.), the procedure to test for significance is the same.

Decide on the critical alpha level we will use (i.e., the error rate we are willing to accept).

1. Conduct the research.
2. Calculate the statistic.
3. Compare the statistic to a critical value obtained from a table.

If our statistic is higher than the critical value from the table:

Our finding is significant.
We rejected the null hypothesis.

The probability is small that the difference or relationship happened by chance, and p is less than the critical alpha level (p < alpha).
If our statistic is lower than the critical value from the table:
Our finding is not significant.
We failed to reject the null hypothesis.
The probability is high that the difference or relationship happened by chance, and p is greater than the critical alpha level (p > alpha).
Modern computer software can calculate exact probabilities for most test statistics. If we have an exact probability from computer software, we will simply compare it to our critical alpha level. If the exact probability is less than the critical alpha level, our finding is significant, and if the exact probability is greater than our critical alpha level, our finding is not significant. Using a table is not necessary when you have the exact probability for a statistic.

This will help determine whether we have a good enough reason to continue the research.

Conclusion/Results

During the course of this project we had various questions in mind- will this app work, is our research apt and most importantly, will we actually make a difference?

This last question kept us going and made sure we didn't loose hope erstwhile the project. Sardar Bhagwant Singh Project gave us a chance to actually explore this problem that persists in 5% kids all over the world. DCD or "clumsy child disease" as many people may call it, is actually a really serious issues that needs immediate attention. We were proud that our work cam actually make a difference but on the other hand, also saddened by the fact that enough resources weren't available to us for our research. This is also another reason why we want to continue this project even after our submission- To raise awareness.

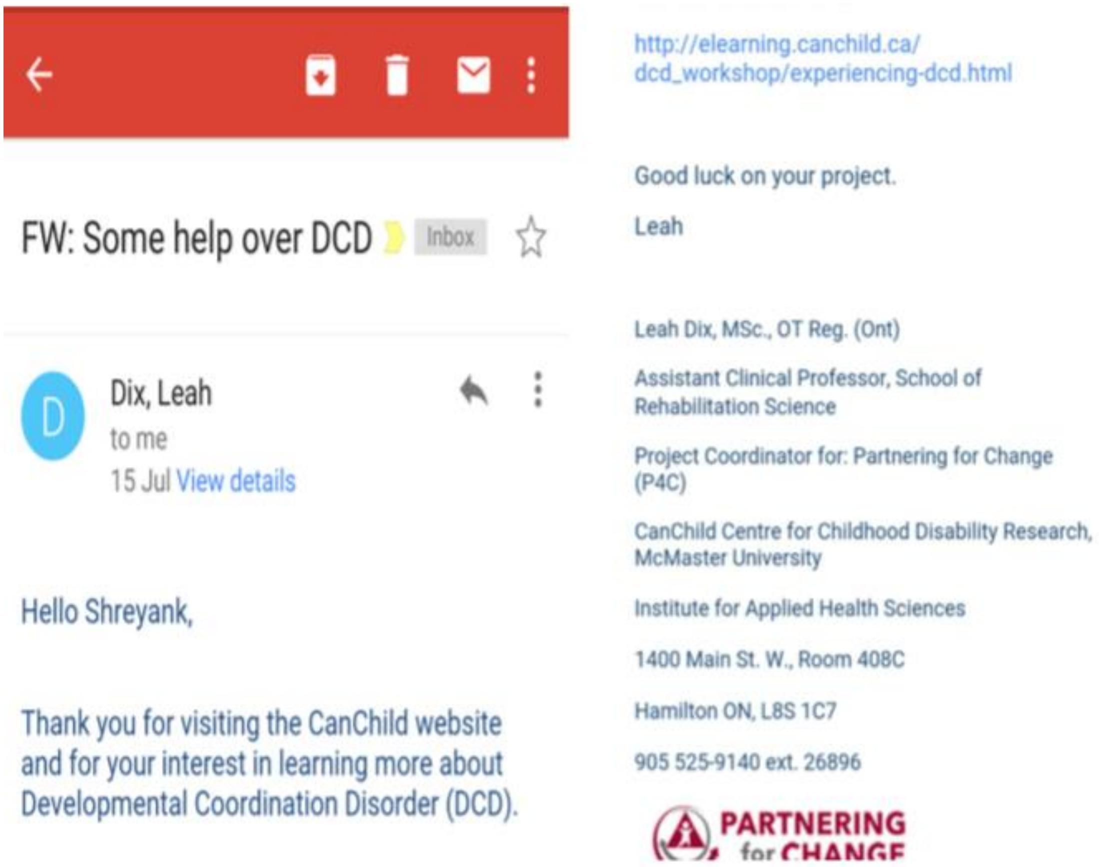
The presented app gives children suffering from this problem an interactive solution. It tests their sudden hand eye coordination thus improving their motor skills.

We believe that this innovative idea can be of major help to this universe. This project aces all guidelines of creativity, social work and cost effectiveness.

In the end we'd again like to shine light on this big big problem that the kids of today are suffering from. Apart from us, doctors from all around the globe are trying to sort it out for children of young age. As we mentioned before DCD persists in 5% of the children all around the globe and even if we manage to help 0.1%, it would be more than enough. Mahatma Gandhi said "Be the change you wish to see in the world" and that being said, we think we're on the right track.

We would also like the emphasise the fact that this project was recognised by the following :-

1. Can child foundation, California
Prof. Leah Dix (MSc. ,OT Reg.) Is an assistant clinical professor at school of rehabilitation science, at "Can Child" foundation seemed to show interest in our project. He provided us with immense resources and made sure there was no flaw in our project. He was happy to see that young kids as ourselves were willing to help other children of our age.
Attached are some screenshots of conversations with Leah over mail



FW: Some help over DCD Inbox

Dix, Leah
to me
15 Jul View details

Hello Shreyank,

Thank you for visiting the CanChild website and for your interest in learning more about Developmental Coordination Disorder (DCD).

http://elearning.canchild.ca/dcd_workshop/experiencing-dcd.html

Good luck on your project.

Leah

Leah Dix, MSc., OT Reg. (Ont)

Assistant Clinical Professor, School of Rehabilitation Science

Project Coordinator for: Partnering for Change (P4C)

CanChild Centre for Childhood Disability Research, McMaster University

Institute for Applied Health Sciences

1400 Main St. W., Room 408C

Hamilton ON, L8S 1C7

905 525-9140 ext. 26896

PARTNERING for CHANGE

2. Frank Anthony Public School, Lajpat Nagar, New Delhi

Our project won the first prize in the event of mathematical modelling. The judges were impressed by the creativity and social use of the project. They were glad to see our presentation and appreciated us during the prize distribution ceremony. They also seemed really interested in the app and were happy to hear about the mathematical principles involved.

This idea bagged the first prize.

Below is a picture of the award:-

3. Skill for action

Skill for action is an online venture started by Pam Versfeld, who's a physiotherapist in private practice from Cape Town, South Africa. Pam believes that all children have the capacity to improve. She helped us in our research and was very friendly in making us understand the conditions of a child suffering from DCD. Pam frequently organised workshops for children suffering from such motor coordination disorders.

Bibliography/ References

CS193P ~ Stanford's iOS app development OCU Professor: Paul Hegarty

Can Child California ~ Developmental Coordination Disorder: Symptoms, Causes and Cures.

MIT iOS App Physics OCU

Sciencebuddies.org ~ How to prepare a scientific project

Harvard Psychology OCU ~ Anatomy of the brain

Wikipedia.com- Developmental coordination Disorder

Acknowledgement

We would like to thank the entire team of the Sardar Bhagwant Singh Project for giving us an opportunity to take up this research project. We would also like to thank Ms. Puja Kapoor for guiding us through it. We appreciate the efforts of Leah Dix and Pam Versfeld in ensuring that this project is successfully made.