GamifyHealth: A Generic Software Framework for Health Behavioral Change

Grace Lee(✉) and Christine Julien

Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, USA
{gracewlee,c.julien}@utexas.edu

Abstract. Many serious games integrate sensed health data to drive behavior change. However, game developers create each game individually, leading to fragmentation, where a new game is created for each new combination of health data and game story. We present GamifyHealth, a generic software framework that integrates sensors for measuring health data with games that encourage health behavior change. We start by identifying elements common to apps that gamify health across three categories: (1) the sensors that provide data; (2) gamification elements; and (3) the mapping of health data to gameplay. Based on a characterization of existing games, we create the GamifyHealth framework that supports generic and flexible development of games for health behavior change. A game can access available health data through a series of observer interfaces provided by the framework. This allows separation of code that interacts with specific sensors and code that presents the game story so that sensor developers and game developers can work independently. Because GamifyHealth provides a clear abstraction of connections from health data to gameplay, a health domain expert can focus attention on connections between sensor data and the game elements that effect behavior change. We have implemented the framework as an Android library and demonstrate its usefulness by re-implementing two existing sensor controlled digital games using GamifyHealth. GamifyHealth simplifies the code of these games and separates the logic for the game developer, the health domain expert, and the sensing integrator, making the resulting implementations more flexible and maintainable.

Keywords: Software framework · Serious games · Sensing

1 Introduction

In recent years, gamification in healthcare has been gaining momentum [19], and previous research has shown that serious games have positive effects in promoting healthy lifestyles [11]. Gamification is defined as the use of concepts that are typical of gameplay in situations that are not strictly games [4]. According to a report from IQVIA Institute for Human Data Sciences, there were over 300,000

H. Lewy and R. Barkan (Eds.): PH 2021, LNICST 431, pp. 435–451, 2022.
https://doi.org/10.1007/978-3-030-99194-4_27

mHealth applications available in app stores as of 2017 [1], many of which have some aspect of gamification. These gamified health apps are in almost every field of health care, ranging from apps that promote physical fitness [15], to those for medication adherence, chronic disease care, or mental health management. These apps have audiences that include people of all ages. Generically, these apps use real-time health data to influence game play. By connecting progress or points in a game to positive health behaviors as indicated by the health data, the apps aim to motivate positive health behavior change. The key components of any such app are therefore the health data sources (which are commonly sensors or self-reports from an individual), the game story and elements of game play, and the health domain knowledge that connects the data to the game.

Unfortunately, the gamified health app space is very *fragmented*, a term we use to refer to the fact that games are developed independently, and thus it is difficult to reuse code, despite the fact that apps share similar components. When the gamified app designs are completely independent, each designer determines (1) which health sensors are used to collect data; (2) which game elements and what game story are used to motivate behavior; and (3) how health data is connected to game elements. If an individual uses more than one gamified health app, the apps operate independently, from collecting data from independent sensor streams to employing motivational tools that are specific to the game design but not specific to the individual. To the best of our knowledge, there is no software framework that supports generic gamification of apps for health behavior change in a way that (1) allows sensor streams to be connected to multiple games and (2) allows different users to have entirely different game stories or game play elements for the same health behavior change purpose.

We have developed GamifyHealth, a generic software framework that enables the development of health behavior change games based on real-time sensor data at higher levels of abstractions. When using GamifyHealth, game developers and health domain experts do not have to deal with the user's health data streams directly. Health sensing experts can focus on the best ways to get real-time sensor information into a consumable format, game development experts can focus on essential aspects of game play, and health domain experts can focus on how health behavior changes can result from connecting sensor data to game elements. If multiple apps rely on the same sensed data, the same data streams can be reused. If a set of game elements that make up a game can be driven by different health data, they can be repurposed.

To enable this flexibility, Gamifyhealth provides abstractions of the common elements found in these three principle components of gamified health behavior change apps: acquiring health data from sensors, game elements that underlie serious games, and the connection between sensor data and game elements that realizes health behavior change. Our key contributions are as follows:

– To create a generic framework, we first identified common components in serious games, health data that is related to these games, and goals that the patient should achieve in order to promote health behavior change.

– We implemented the GamifyHealth framework as an Android library, which enables generic connections from real-life sensors to arbitrary game elements.
– We used the Android library implementation of the GamifyHealth framework to mock the reimplemenation of two existing sensor controlled health games.

2 Related Work

Recently, the healthcare domain has been using serious games to motivate and promote healthy behavior [23] in various domains, including heart failure [20], diabetes [14], and treatment adherence [17]. It has been shown that gamification has positive impact on self-managing health behavior [3,11].

For instance, Acticore [24] helps users learn about and train their pelvic floor muscles. A user sits on a sensor seat that connects to the app via Bluetooth. With the health data provided from the seat, Acticore displays an avatar and engages the user in games that train different aspects of the pelvic floor muscles.

FunSpeech [5] motivates children who were born deaf to practice their speech. The app takes audio as input and provides immediate feedback through mini-games that address different aspects of speech, such as pitch, intensity, and phoneme construction. In one mini-game, the user controls a helicopter with the pitch of their voice. In another, the player makes monkeys disappear by producing sounds at a specific volume for a specific duration of time.

Richard et al. [22] developed an underwater exploration game that motivates people affected by cystic fibrosis to perform airway clearance exercises. An incremental pressure sensor is attached to a Positive Expiratory Pressure device to detect the user's breathing. When the user employs correct breathing patterns as instructed, they can "swim" forward in the game, and they are also rewarded with treasures upon reaching a target "diving distance".

Heart Mountain [20] targets older patients with chronic heart diseases. Each user has an activity sensor and a smart scale to track step count and whether they weighed themselves on a daily basis. By achieving a daily step count goal and weighing in every day, the user earns coins and makes progress in the game.

Without a generic framework, each of these serious games is developed independently. This makes it difficult to personalize games to fit each user's preferences. For example, if a person with cystic fibrosis would rather control helicopters than explore underwater, a different app would have to be developed, although the code for the helicopter game and breathing sensors have already been developed by existing teams. With a generic framework, the sensor portion of the app and the game elements would be decoupled, allowing gamified health apps to be tailored towards each user without increasing workload of developers.

Multiple prior efforts have provided different lists of what are considered common elements in gamification. Garett and Young [7] provided a list of some common gamification elements in health care, such as points, leaderboards, progress status, and badges/medals. In this paper, we took the common gamification elements and grouped them based on the similarities of their structure and purpose. In other words, we abstract the commonality of the different but similar elements into components of the software framework.

There have been a few other software frameworks or middlewares aimed at supporting digital interventions using sensor data or gamification. For instance, Koutsouris et al. developed a platform that merges the user's activity in real life with a real game that they play in [16]. UNITY-Things [25] is a software framework specifically designed to link Arduino-enabled devices with the UNITY game engine, instead of supporting connections between generic devices and games.

Göbel et al. [8] designed a technical framework for exergames that also allows the integration of sensors into the game. They also provide a user interface that allows health domain experts to configure the game to fit the user's health needs. In contrast, our work focuses on a different problem; GamifyHealth is a software framework that provides generic functionalities, as well as abstract components for developers to plug in their code to create any sensor-connected game they envision.

The Ayogo Model [13] explicitly incorporates health domain experts into the game design process; similar to our software framework, in the Ayogo Model, health domain experts identify health behavior goals that are tailored towards a chronic illness. However, our contributions are focused on how to simplify the process of developing a full stack game, whereas the Ayogo Model focused on how to design and develop a game that taps into a user's ability to self manage.

3 The GamifyHealth Framework

We next describe our contributions via the GamifyHealth framework in two stages. First, we explore existing serious games for health behavior change to identify common components connecting real-time sensing, game elements, and health behavior change. Using these constructs, we then present the design of the GamifyHealth framework and its application programming interfaces (APIs).

3.1 Common Components in Games for Health Behavior Change

To identify generic components across games for health behavior change, we studied several games, including (1) health behavior change related games from the six editions of the Gamification & Serious Game Symposiums that include mobile games and sensors[1]; (2) games published in other conferences or journals; and (3) some popular games available in app stores. We first provide an overview of the 13 games we reviewed, then look at them across our three dimensions.

- Ludicross [2]: the player's forced exhalations, measured by a spirometer, drive a virtual car around a race track.
- FunSpeech [5]: the app's collection of mini-games all take sound as input; the player is asked to make or sustain sounds to achieve game goals.

[1] https://gsgs.ch/.

- Mission:Schweinehund [10]: in this gardening-themed game, upon achieving daily step count goal or completing in-game physical activities, the user gets materials to help tend and grow a virtual garden.
- Project SMART [12]: elementary school students move along a virtual journey by being physically active as measured by accelerometers or self-reports.
- Monster Manor [13]: children log their blood glucose to earn virtual currency that they use to purchase virtual items needed to progress through the game.
- Type 2 Travelers [13]: players log medication and glucose monitoring activities to earn virtual currency to personalize an avatar and unlock new game levels; the game also includes social interaction with other patients.
- Picture It [13]: this game integrates an activity monitor and helps promote weight loss and healthy habits in patients preparing for bariatric surgery.
- Hand washing game for nurses [18]: sensors on a sink or alcohol dispenser measure time spent near these areas to determine if users complied to hand hygiene standards. Each user's compliance is shown in a leaderboard.
- Heart Mountain [20]: users climb a virtual mountain and earn coins and awards by achieving a daily step count and weigh in; the player's avatar dynamically reflects the player's success in the game's healthy habits.
- Underwater exploration [22]: users can swim forward to explore underwater in the game and earn treasures by doing airways clearance exercises.
- Acticore [24]: an avatar cheers the user on and provides feedback for doing pelvic floor muscle exercises on a "Sensor Seat".
- Stay Fit Longer [26]: by completing a set of daily quests involving physical and cognitive activities, users can earn daily rewards.
- Stellar Spine: scoliosis braces for children are outfitted with a Bluetooth-enabled sensor; achieving a prescribed number of hours of wear time daily allows the child to collect stars and eventually construct constellations.

We organized the common components into three categories: components for sensing health data, game elements, and components that bridge the two.

Integrating Sensing of Health Data in Serious Games. A wide variety of sensors are integrated into games in order to encourage health behavior change. Depending on the purpose of a game, the health data may be collected in a variety of ways: data may be pulled directly from wearable devices or sensors in a user's home; it may be retrieved from a cloud server that belongs to another service, or the data may be input manually by the user. Table 1 lists the 13 games we reviewed and indicates their mechanism for integrating sensor information.

Identifying Game Elements for Health Behavior Change. In our review of existing games, we identified several key constructs that are commonly connected to sensed health data in games. These include some mechanism for assigning "points" in the game, displaying leaderboards that engender healthy competition, mechanisms for indicating progress towards some goal, and badges or other collectibles that indicate achievement [7]. In addition, games commonly involve some story line and potentially random challenges [9].

Table 1. Sensing integration and gamification elements in studied games

Game	Direct Sensor	Cloud Service	User Input	Goals	Progress	Points	Leaderboard	Avatar
Ludicross [2]	✔			✔				
FunSpeech [5]	✔			✔		✔		
Mission:Schweinehund [10]	✔			✔	✔	✔		
Project SMART [12]	✔		✔	✔	✔		✔	
Monster Manor [13]			✔	✔	✔	✔		
Type 2 Travelers [13]			✔	✔	✔	✔		✔
Picture It [13]		✔	✔	✔		✔		✔
Hand washing game for nurses [18]	✔						✔	
Heart Health Mountain [20]		✔		✔	✔	✔	✔	✔
Underwater exploration game [22]	✔		✔			✔		
Acticore [24]	✔			✔	✔			
Stay Fit Longer [26]			✔	✔		✔	✔	
Stellar Spine	✔			✔		✔		

Connecting Health Data to Game Elements. Serious games for health behavior change also have health goals that guide users to achieve an objective identified by a healthcare professional. For the games in our review, users commonly have to complete goals in real life in order to push the game forward, or receive awards. In other words, goals integrate live health data to game elements. Table 2 shows the goal mapping for each of the games in our review.

Many goals also have a time aspect. For instance, one FunSpeech's mini-games asks the user to produce specific sounds for a given length of time [5]. The underwater exploration game instructs users to exhale continuously for a specific duration [22]. Users earn rewards by doing daily tasks in Heart Mountain [21].

3.2 The Design of the GamifyHealth Framework

A software framework provides generic functionalities that allow developers to plug in their code to create personalized programs. As such, a framework implements some concrete components but leaves others as abstract, to be reified by developers who use the framework. In GamifyHealth, we envision three categories of developers as shown in Fig. 1: (1) sensing integrators; (2) game developers; and (3) health domain experts. These three types of developers have decidedly different areas of expertise and different interests in the resulting game.

Integrating Sensing of Health Data in Serious Games. Building on the mechanisms identified above, GamifyHealth supports three modes of data inte-

Table 2. Connecting health data to game goals

Game	Example Data → Goal
Ludicross [2]	perform measured forced exhalation → make car progress around a race track
FunSpeech [5]	emit a sound at the specified time → earn fish in game
Mission:Schweinehund [10]	complete in-game workouts → earn materials in game storyline
Project SMART [12]	physical activity → progress on virtual journey
Monster Manor [13]	input blood glucose → earn coins
Type 2 travelers [13]	log medication and glucose monitoring → earn coins, unlock levels, interact with others
Picture it [13]	achieve activity and adhere to healthy goals → update avatar and earn currency
Hand washing game for nurses [18]	achieve hand hygiene compliance → move up leaderboard
Heart mountain [20]	achieve step goal and weigh in → advance game levels and earn coins
Underwater exploration game [22]	correctly complete breathing exercise → advance in game progress
Acticore [24]	complete training schedule → avatar comments on the performance
Stay fit longer [26]	complete daily quests → earn coins
Stellar spine	achieve daily brace wear-time goal → earn stars and learn about constellations

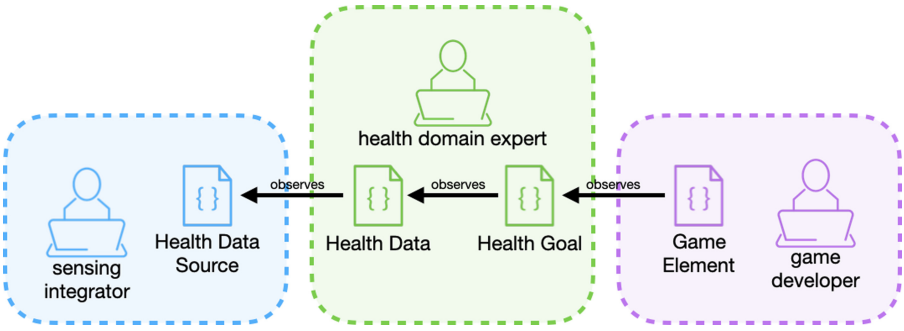

Fig. 1. Separation of concerns in GamifyHealth by developer expertise

gration: direct sensing, data collection from a server, and manual user input. All three of these mechanisms are unified under a single abstract `HealthDataSource` component in the framework, shown in Listing 1 (lines 2–12).

Listing 1. Sensing Integration Interfaces

```
1  package com.example.healthgamifylib
2  abstract class HealthDataSource(val name: String) : Subject() {
3    var value: Any = -1
4    override fun notifyObservers() { for (o in observers) { o(value) } }
5    // template method implements 2 step algorithm: update the value and notify observers
6    fun updateValue(): Unit {
7      updateHealthDataFromSource()
8      notifyObservers()
9    }
10   // this method is overridden by sensor integrator
11   abstract fun updateHealthDataFromSource() : Unit
12 }
13 abstract class Subject {
14   var observers: MutableList<(Any?) -> Unit> = mutableListOf()
15   fun registerObserver(whatToCall: (Any?) -> Unit) : Unit {
16     observers.add(whatToCall)
17   }
18   fun removeObserver(whatNotToCall: (Any?) -> Unit) : Unit {
19     observers.remove(whatNotToCall)
20   }
21   abstract fun notifyObservers() : Unit
22 }

24 import com.example.healthgamifylib.HealthDataSource
25 class MyDataSource(name: String) : HealthDataSource(name){
26   override fun updateHealthDataFromSource(){
27     // sensor-specific implementation written by sensor integrator
28   }
29 }
```

Each `HealthDataSource` exposes a `value` that stores the value of the collected data. The `HealthDataSource` component relies on the *Observer* pattern [6], where the health data source itself is an observable component, and other components in the game (the observers) can observe changes to the data value stored in the data source. To realize this in the GamifyHealth framework, the `HealthDataSource` extends the `Subject` abstract class (shown in Listing 1 in lines 13–22), which manages the registered observers. The `HealthDataSource` implements a *template method* called `updateValue` (lines 6–9 in Listing 1) which retrieves the updated health data from the data source and then notifies any registered observers. The `updateValue` method is final so that derived classes cannot override it and change the template.

The developer integrating health sensing into a game constructed with the GamifyHealth framework does not need to concern themselves at all with the observation of the value. Instead, the sensor integrator simply does the following:

- create a class derived from `HealthDataSource` for the specific sensor
- as part of fulfilling the contract of the extension, write the implementation for `updateHealthDataFromSource` (for example, as lines 24–29 in Listing 1)
- define how `updateValue` in the new `HealthDataSource` is called, whether periodically on a timer or triggered by new data available from a sensor, etc.

Game Elements for Health Behavior Change. The second type of developer is the game developer, responsible for the user's perception of the game

Listing 2. Game Developer Interfaces

```
1  package com.example.healthgamifylib
2  abstract class Points (var maxValue: Int) : Observer {
3    var value: Int = 0
4  }
5  interface Observer {
6     fun update(value: Any?) : Unit
7  }

9  import com.example.healthgamifylib.Points
10 class MyPoints(maxValue: Int) : Points(maxValue) {
11   override fun update(value: Any?) {
12     // game-specific implementations of what happens when points change
13     // written by the game developer
14   }
15 }
```

Listing 3. Health Domain Expert Interfaces: The Health Data Class

```
1  package com.example.healthgamifylib
2  open class HealthData(val healthDataSource: HealthDataSource) : Subject(), Observer {
3    var value: Any = -1
4    var timestamp: Date? = null
5    open override fun update(value: Any?) {
6      this.value = value as Int
7      this.timestamp = LocalDateTime.now()
8    }
9    override fun notifyObservers() { for (o in observers) { o(value) } }
10 }
```

play. A significant portion of this person's effort is focused on the visual and story elements of the game; the GamifyHealth framework allows this person to not have to consider health sensing details. GamifyHealth supports a variety of game elements that can observe the health goals defined by the health domain expert in Fig. 1 (e.g., a daily step goal or a wear-time goal or a forced exhalation target). We describe the implementation of these goals in the next subsection. However, as one example of such a game element, consider the concept of `Points`. Listing 2 shows the abstract `Points` class, which is an observer of some other component (i.e., a goal). The game developer simply provides a definition of the `Observer`'s `update` method; every time the observed goal's value changes, the `update` method is called, and the associated element in the game also changes, based on the developer's definition of the method. Ultimately, this `update` method (e.g., line 11 in Listing 2) is passed as the observer to the observed goal's `registerObserver` method. This connection is made by the health domain expert, who connects health data sources to game elements. These processes are described next. In addition to these `Points`, game developers will implement progress measures, leaderboards, and the other game elements we identified previously. Their implementations mirror that of `Points`.

Connecting Health Data to Game Elements. The health domain expert is responsible for: (1) health data, which abstracts away from the specific sensors that collect data and (2) health goals, which expose hooks for gamification. A `StepCount` class might be an instance of `HealthData`, while a `FitBit`

or a `WithingsTracker` may be a `HealthDataSource` that can be observed by `StepCount`. The `HealthData` component (Listing 3) is a simple observer of a `HealthDataSource`. `HealthData` provides a simple implementation of the `update` method in the `Observer` contract. However, this class is also open for extension if a derived `HealthData` needs to convert from a lower level sensed data type into a higher level one. For instance, in Stellar Spine, the sensor captures body temperature, which is converted into a boolean indicating whether the brace is being worn or not, depending on a comparison to expected body temperature.

The `HealthData` component is also observable; in particular, it is observed by goals. To show how GamifyHealth supports defining goals, we provide examples of the `WindowGoal` and `RepeatingWindowGoal` in Listing 4. These directly respond to the fact that many goals in the games we reviewed are associated with completing a task for a duration (e.g., a sustained forced exhalations) or within a duration (e.g., achieving a daily step count).

Each `WindowGoal` (lines 2–35 in Listing 4) has a `window` to store the length of the duration and a `goalAchieved` to store whether the goal has been achieved within the window. For example, line 10 of Listing 4 checks whether the `value` of the embedded `HealthData` object has reached or exceeded the `targetValue`. This is a common pattern seen in existing games, but a health domain expert can also extend `WindowGoal` and override the `update` method to provide a different logic for evaluating reaching the goal.

Upon creation, a `WindowGoal` schedules a `StartWindow` task and an `EndWindow` task. `StartWindow` sets `goalAchieved` to `false` and registers as an observer of the `HealthData`. `EndWindow` unsubscribes from the `HealthData` and calls `finalizeGoal` (line 16), whose main purpose is to notify the `WindowGoal`'s observers when the player fails to achieve the goal (when the player achieves the goal, any registered observers are notified immediately). The developer can also optionally override the `finalizeGoal` method.

To support goals that repeat, GamifyHealth uses `RepeatingWindowGoal`. Its `repetitions` variable stores the number of `WindowGoal`s to create, `streak` stores the target streak the player is aiming to achieve, and `currentStreak` stores the number of consecutive achieved goals, as calculated upon the end of each `WindowGoal` (lines 49–53). For example, if the player needs to complete a daily goal for a week, but misses the goal on the 5^{th} day, then `repetitions` would have a value of 7, and after the 7^{th} day ends, the `currentStreak` is 2 (days 6 and 7). To make sure that the previous `WindowGoal` closes before a new `WindowGoal` is created, `RepeatingWindowGoal` passes a `ReentrantLock` to each `WindowGoal` (lines 65 and 59). The `WindowGoal` locks the `ReentrantLock` at the beginning of `StartWindow` and unlocks at the end of `EndWindow`.

For a `Points` classes to observe a `RepeatingWindowGoal`, the game developer registers `Points`'s `update` method as an observer of `RepeatingWindowGoal`. The `RepeatingWindowGoal` registers the observers to each of the `WindowGoal`s it creates (Listing 4, lines 60 and 66). The `notifyObserver` method updates its observers of the `currentStreak` when `streak` is achieved (Listing 4, line 51).

Listing 4. Health Domain Expert Interfaces: The Health Goal Classes

```
package com.example.healthgamifylib
open class WindowGoal(var targetValue: Int, var observedData: HealthData,
                      var start: Date, var window: Duration, val lock: ReentrantLock) :
                            Subject(), Observer {
  private val timer = Timer()
  var goalAchieved: Boolean = false
  // satisfying the Subject contract
  override fun notifyObservers() { for (o in observers) { o(goalAchieved) } }
  // satisfying the Observer contract; can be overridden for more tailored behavior
  open override fun update(value: Any?){
    if(!goalAchieved && value >= targetValue) {
      goalAchieved = true
      notifyObservers()
    }
  }
  // finalizeGoal can be overridden if player is not penalized for not achieving the goal
  open fun finalizeGoal() { if(!goalAchieved) { notifyObservers() } }
  private inner class StartWindow() : TimerTask() {
    override fun run() {
      lock.lock()
      goalAchieved = false
      observedData.registerObserver(this@WindowGoal::update)
    }
  }
  private inner class EndWindow() : TimerTask() {
    override fun run() {
      observedData.removeObserver(this@WindowGoal::update)
      finalizeGoal()
      lock.unlock()
    }
  }
  init {
    timer.schedule(StartWindow(), start)
    timer.schedule(EndWindow(), start + window)
  }
}
open class RepeatingWindowGoal(targetValue: Int, observedData: HealthData,
                               start: Date, window: Duration,
                               var repetitions: Int, var streak: Int = 0) : Subject() {
  var embeddedWindowGoal : WindowGoal
  val timer: Timer = Timer()
  val goalArray = mutableListOf<Boolean>()
  var repetitionsCompleted = 0
  val lock = ReentrantLock()
  var currentStreak = 0
  override fun notifyObservers() { for (o in observers) { o(streak) } }
  private inner class UpdateWindowGoal() : TimerTask() {
    override fun run() {
      goalArray.add(embeddedWindowGoal.goalAchieved)
      if(emeddedWindowGoal.goalAchieved) {
        currentStreak++
        if (currentStreak >= streak){ notifyObservers() }
      }
      else { currentStreak = 0 }
      notifyObservers()
      repetitionsCompleted++
      if(goalArray.size < repetitions) {
        newStart = start + goalArray.size * window
        embeddedWindowGoal = WindowGoal(targetValue, observedData, newStart, window, lock)
        for (o in observers) { embeddedWindowGoal.registerObserver(o) }
      }
    }
  }
  init { // this creates and starts the first repetition of the window goal
    embeddedWindowGoal = WindowGoal(targetValue, observedData, start, window, lock)
    for (o in observers) { embeddedWindowGoal.registerObserver(o) }
    timer.schedule(UpdateWindowGoal(), start + window, window)
  }
}
```

4 Case Study on the Framework

We have implemented a prototype of our framework design as an Android library[2]. We re-implemented two of the games from our review to use this framework. In these mock implementations, we do not focus on the graphical interfaces associated with game play and instead focused on implementing the pipeline shown in Fig. 1. We seek to answer the following research questions:

- How do the GamifyHealth generic constructs manifest themselves in real games for health behavior change?
- Can the GamifyHealth framework be used to integrate real-world sensing into games for health behavior change?
- Do the GamifyHealth framework representations of health goals support the needs of real games?

4.1 Case Study Game 1: Heart Mountain

Heart Mountain [21] is a sensor controlled digital game for patients with heart failure. Players are challenged to complete a daily step goal, capture their weight with a digital scale, and progress by answering quizzes about heart failure.

Integrating Sensing of Health Data in Heart Mountain. Each player has two physical sensors: an activity tracker and a smart scale, both of which are connected to the Withings server. The game downloads each player's data from the Withings server. Listing 5 shows the `WTActivityTracker` class that the Heart Mountain sensor integrator would write. Upon initialization, the sensor integrator is responsible for authenticating the user with their Withings account (e.g., by showing a login modal). Then the sensor integrator defines an update period for the data; when the timer fires, the task calls the base class's `updateValue` method, which in turn calls `updateHealthDataFromSource` to retrieve the data from the Withings API, followed by a call to notify any observers.

Game Elements for Health Behavior Change in Heart Mountain. For the game developer, the most important elements are (1) the user's points, in terms of both hearts and coins; (2) the user's avatar, whose appearance changes based on the player's collection of hearts; and (3) the player's progress up the heart mountain. Hearts and coins are straightforward extensions of the `Points` class, with simple implementations of the `update` method. The avatar is squarely in the game developer's expertise and can be updated based on updates to the hearts value. Finally, progress in the game can be inferred from an array of accomplished goals that the game developer can easily create and store. The majority of the work of the game developer for the Heart Mountain game would be focused on the user interface and graphics associated with the user experience.

[2] https://github.com/UT-MPC/HeartHealthMountain-Kotlin/tree/pervasiveHealth.

Listing 5. Heart Mountain: Sensing Integration

```
package com.example.heartmountain
class WTActivityTracker(name: String, val updateInterval: Int) : HealthDataSource(name) {
  inner class NewValue(): TimerTask(){
    override fun run() {
      updateValue()
    }
  }
  override updateHealthDataFromSource(){
    /** make API call to Withings server to retrive step data **/
    value = /** retrieved step data **/
    timer.schedule(NewValue(), LocalDateTime.now() + updateInterval)
  }
  init {
    /** authenticate user to Withings backend **/
    val timer = Timer()
    timer.schedule(NewValue(), LocalDateTime.now() + updateInterval)
  }
}
```

Listing 6. Heart Mountain: Connecting Sensing to the Game

```
package com.example.heartmountain
class MainApplication() : Application() {
  override fun onCreate() {
    // update interval of one day
    val updateInterval = 86400
    // create a type of Points, called Heart, with maxValue 1000
    val heart = Heart(1000);
    // create the withings scale data source
    val withingsScale = WTScale(name = "Withings Scale", updateInterval = updateInterval)
    // create a HealthData around the Withings scale data source
    val weighed = HealthData(healthDataSource = withingsScale)
    // create a repeating window goal for the daily weigh in
    val dailyWeighIn = RepeatingWindowGoal(targetValue = 1, observedData = weighed,
                                           start = startDate, window = updateInterval,
                                           repetitions = 30, streak = 25)
    // register the game construct heart as an observer of the goal
    dailyWeighIn.registerObserver(heart::update)
    /** something similar for the step goal **/
  }
}
```

Connecting Health Data to Game Elements in Heart Mountain. The health domain expert plays a pivotal role in connecting health behaviors as measured by data sources to game elements that motivate health behavior change. In Health Mountain, in particular, the health domain expert needs to define two types of `HealthData`: (1) daily steps and (2) daily weigh-in. They would then also define the `RepeatingWindowGoal`s the player is expected to achieve, i.e., hitting a target number of steps and simply stepping on the scale each day.

However, the health domain expert is not expected to be an expert programmer, and the GamifyHealth framework seeks to ease their programming burden as much as possible. In many cases the health domain expert does not need to extend the framework but rather only need instantiate instances of existing `HealthData` and `WindowGoal` classes. Listing 6 shows an example of creating these connections in the Heart Mountain game.

Listing 7. Stellar Spine: Game Developer Interfaces

```
package com.example.stellarspine
import com.example.healthgamifylib.Points
class Stars(maxValue: Int) : Points(maxValue) {
  // each constellation is a variable number of stars
  val constellations = /** collection of constellations **/
  var constellationsCollected: Int = 0
  var stars: Int = 0
  var starsUsed: Int = 0
  override fun update(value: Any?) {
    val braceWorn = value as Boolean
    if(braceWorn) {
      stars++
      val starsAvailable = stars - starsUsed
      if(starsAvailable >= constellations[constellationsCollected].stars) {
        starsUsed += constellations[constellationsCollected].stars
        constellationsCollected++
      }
    }
  }
}
```

4.2 Case Study Game 2: Stellar Spine

Stellar Spine is designed to encourage children with scoliosis to wear a brace for a certain amount of time per day. Stellar Spine uses Bluetooth to read data from a temperature sensor embedded in a brace that the child wears around their chest. The temperature sensed is used to determine if the brace is worn or not, based on the value's similarity to expected body temperature. In Stellar Spine, players earn stars for every day they reach their target wear time. These stars construct constellations, and players learn facts about the stars and constellations.

Integrating Sensing of Health Data in Stellar Spine. The sensor integrator for Stellar Spine needs to create a `HealthDataSource` class that sets up Bluetooth bonding with the device embedded in the brace. The design of the Bluetooth connection for Stellar Spine was intentional with respect to ensuring battery longevity for the sensor in the brace; therefore the update method is triggered manually via a button on the brace combined with a simultaneously performed explicit refresh in the app. As a result, instead of updating the `HealthDataSource` periodically, the `updateValue` method needs to be triggered explicitly from the game's API. Once triggered, the base class's `updateValue` method can call the derived class's `updateHealthDataFromSource`, which can then retrieve the data from Bluetooth and format it into an array of temperature values. These correspond to the average temperature sensed in 15 min intervals since the last retrieved update. This array is stored in the `HealthDataSource`'s `value`.

Game Elements for Health Behavior Change in Stellar Spine. The primary game elements for Stellar Spine are the stars, for which we created a class `Stars` that derives from `Points`. It stores the number of stars the user has earned, as well as the constellations that the stars form. Listing 7 shows

Listing 8. Stellar Spine: Health Domain Expert Interfaces

```
package com.example.stellarspine
import com.example.healthgamifylib.HealthData
class WearTime(source: HealthDataSource) : HealthData(source) {
  val bodyTemp = 37
  override fun update(value: Any?){
    // value from source is array of temperature values
    val valueAsArray = (value as? Array<*>)?.filterIsInstance<Int>()
    var sum: Int = 0
    for(i in valueAsArray) {
      if (bodyTemp - 1 < i < bodyTemp + 1) { sum = sum + 1 }
    }
    // each temperature represents a 15 minute window
    value = sum * 15
  }
}
```

the mock implementation for this class. When `update` is called, if the value propagated from the daily wear goal is `true`, the player collects a new star. Subsequently, if this new star means the player has collected enough stars for the next constellation, the data structures are updated. The game developer can use this information to update the game interface, e.g., displaying the new constellation.

Connecting Health Data to Game Elements in Stellar Spine. The Stellar Spine mock game has two details that fall in the purview of the health domain expert: (1) converting temperature values to wear time values in a `HealthData` class and (2) defining the repeating wear time goal. To solve the first, the health domain expert creates a class derived from `HealthData` called `WearTime` that observes the brace sensor's `HealthDataSource`. If the temperature sensed is close to body temperature, `WearTime`'s value is incremented. Because the brace sensor's `HealthDataSource` contains an array of temperature values, this converts to an array of times, measured in 15 min intervals. This is shown in Listing 8.

The health domain expert must also create a repeating window goal that has a target wear time equivalent to the "prescription" given to the player. If the player achieves that amount of wear time in a day, then the player achieves the goal and is awarded the daily star (as shown in Listing 7).

5 Conclusions and Future Work

We presented GamifyHealth, a generic software framework that supports flexible development of serious games for health behavior change by providing abstractions in three categories: (1) sensors that provide data that drives the game (e.g., health data); (2) elements of gameplay (e.g., points); and (3) the mapping of health data to gameplay. The separation of these three categories allows sensing integrators, game developers, and health domain experts to work independently, which also allows sensor streams to be connected to multiple games.

We implemented the framework as an Android library. To support other operating systems, a future implementation of the framework could be a multi-platform Kotlin implementation, which would allow GamifyHealth to work on all platforms that Kotlin supports, which includes iOS. We demonstrated the usability of GamifyHealth by creating mock implementations of two existing games, Heart Mountain and Stellar Spine. We are currently extending the mock implementations to develop full implementations of the two games, so that the original code and the code implementing GamifyHealth can be further compared. We are also implementing a hybrid of the two games, i.e., health data used in Heart Mountain connected to the game elements of Stellar Spine, to further explore the potential of GamifyHealth.

References

1. Aitken, M., Clancy, B., Nass, D.: The growing value of digital health: evidence and impact on human health and the healthcare system (2017)
2. Bingham, P.M., Lahiri, T., Ashikaga, T.: Pilot trial of spirometer games for airway clearance practice in cystic fibrosis. Respir. Care **57**(8), 1278–1284 (2012)
3. Charlier, N., Zupancic, N., Fieuws, S., Denhaerynck, K., Zaman, B., Moons, P.: Serious games for improving knowledge and self-management in young people with chronic conditions: a systematic review and meta-analysis. J. Am. Med. Inf. Assoc. **23**(1), 230–239 (2015). https://doi.org/10.1093/jamia/ocv100
4. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining "gamification". In: Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011, p. 9–15. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2181037.2181040
5. Florent, G., et al.: Funspeech: Promoting speech production in young children with hearing disabilities. In: International Conference on Gamification and Serious Game (2019)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software, 1st edn. Addison-Wesley Professional, Boston (1994)
7. Garett, R., Young, S.D.: Health care gamification: a study of game mechanics and elements. Technol. Knowl. Learn. **24**, 341–353 (2018)
8. Göbel, S., Hardy, S., Wendel, V., Mehm, F., Steinmetz, R.: Serious games for health: Personalized exergames. In: Proceedings of the 18th ACM International Conference on Multimedia, MM 2010, pp. 1663–1666. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1873951.1874316
9. Hamari, J., Koivisto, J., Sarsa, H.: Does gamification work? - a literature review of empirical studies on gamification. In: 2014 47th Hawaii International Conference on System Sciences, pp. 3025–3034 (2014). https://doi.org/10.1109/HICSS.2014.377
10. Höchsmann, C., Infanger, D., Klenk, C., Königstein, K., Walz, S.P., Schmidt-Trucksäss, A.: Effectiveness of a behavior change technique-based smartphone game to improve intrinsic motivation and physical activity adherence in patients with type 2 diabetes: Randomized controlled trial. JMIR Ser. Games **7**(1), e11444 (2019)

11. Johnson, D., Deterding, S., Kuhn, K.A., Staneva, A., Stoyanov, S., Hides, L.: Gamification for health and wellbeing: a systematic review of the literature. Internet Interv. **6**, 89–106 (2016)
12. Julien, C., Castelli, D., Bray, D., Lee, S., Burson, S., Jung, Y.: Project smart: a cooperative educational game to increase physical activity in elementary schools. Smart Health **19**, 100163 (2021)
13. Kamel Boulos, M.N., et al.: Digital games for type 1 and type 2 diabetes: underpinning theory with three illustrative examples. JMIR Ser. Games **3**(1), e3390 (2015)
14. Klingensmith, G.J., et al.: Evaluation of a combined blood glucose monitoring and gaming system (Didget®) for motivation in children, adolescents, and young adults with type 1 diabetes. Pediatr. Diabetes **14**(5), 350–357 (2013)
15. Knight, E., Stuckey, M.I., Prapavessis, H., Petrella, R.J.: Public health guidelines for physical activity: is there an app for that? A review of android and apple app stores. JMIR mHealth uHealth **3**(2), e4003 (2015)
16. Koutsouris, N., Kosmides, P., Demestichas, K., Adamopoulou, E., Giannakopoulou, K., De Luca, V.: Inlife: a platform enabling the exploitation of IoT and gamification in healthcare. In: 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 224–230 (2018). https://doi.org/10.1109/WiMOB.2018.8589153
17. Whiteley, L., Brown, L., Lally, M., Heck, N., van den Berg J: A mobile gaming intervention to increase adherence to antiretroviral treatment for youth living with HIV: development guided by the information, motivation, and behavioral skills model. JMIR Mhealth Uhealth. **6,** e8155 (2018). https://doi.org/10.2196/mhealth.8155,https://mhealth.jmir.org/2018/4/e96
18. Lapão, L., Marques, R., Gregorio, J., Pinheiro, F., Povoa, P., Mira da Silva, M.: Using gamification combined with indoor location to improve nurses' hand hygiene compliance in an ICU ward. Stud. Halth Technol. Inform. **221**, 3–7 (2016). https://doi.org/10.3233/978-1-61499-633-0-3
19. Lister, C., West, J.H., Cannon, B., Sax, T., Brodegard, D.: Just a fad? Gamification in health and fitness apps. JMIR Ser. Games **2**(2), e3413 (2014)
20. Radhakrishnan, K., et al.: Abstract 13122: sensor-controlled digital game may improve weight monitoring among older adults with heart failure. circulation **142**(Suppl_3), A13122–A13122 (2020). https://doi.org/10.1161/circ.142.suppl_3.13122
21. Radhakrishnan, K., et al.: Usability assessment of a sensor-controlled digital game for older adults with heart failure. Innov. Aging. **3**, S8192 (2019)
22. Richard, W., Tobias, K.: Prototyping a virtual reality diving game to support breathing exercises to treat cystic fibrosis. In: International Conference on Gamification and Serious Game (2019)
23. Sardi, L., Idri, A., Fernández-Alemán, J.L.: A systematic review of gamification in e-health. J. Biomed. Inform. **71**, 31–48 (2017)
24. Sebastian, I.: Ani: how to turn a muscle into a motivating character. In: International Conference on Gamification and Serious Game (2019)
25. Svanæs, D., Scharvet Lyngby, A., Bärnhold, M., Røsand, T., Subramanian, S.: Unity-things: an internet-of-things software framework integrating Arduino-enabled remote devices with the unity game engine. In: Fang, X. (ed.) HCI in Games: Experience Design and Game Mechanics, pp. 378–388. Springer International Publishing, Cham (2021)
26. Sylvain, C., et al.: Gamification to improve adherence in home-based activities for seniors. In: International Conference on Gamification and Serious Game (2019)