Declarative Physicomimetics for Tangible Swarm Application Development

Ayberk Özgür[1(✉)], Wafa Johal[1,2], Arzu Guneysu Ozgur[1], Francesco Mondada[2], and Pierre Dillenbourg[1]

[1] CHILI, EPFL, Lausanne, Switzerland
{ayberk.ozgur,wafa.johal,arzu.guneysu,pierre.dillenbourg}@epfl.ch
[2] LSRO, EPFL, Lausanne, Switzerland
francesco.mondada@epfl.ch

Emerging interest in exploring proximal Human-Swarm Interaction has started approaching the Human-Computer Interaction (HCI) vision of tangible, bidirectional interaction with intelligent swarms made up of "radical atoms" [2]. The recent years have witnessed significant progress towards these once-hypothetical materials in the form of Tangible Swarm Robots, for which the focus shifted towards developing *applications* for the user to interact with, rather than *controllers* for the robots to solve tasks. Still, how and with which tools HCI designers could build such applications in a swift and reusable manner is an open issue.

We propose here such an application development framework which combines two existing approaches: First, we program swarms with *virtual forces* that describe and create robot motion (*i.e.* physicomimetics [4]) instead of coding individual or collective actions over time. Second, we use the Qt Modeling Language (QML) [1], a declarative programming language originally designed to develop graphical user interfaces by *declaring objects* and *binding* their *properties* and *events* to create the program's structure and flow. Our core idea is to define the swarm of robots and their behaviors (forces, tangible input detectors *etc.*) in terms of these modular and reusable constructs. Below, we provide the program for a rudimentary "bubble shooter" game that illustrates this (details such as calibration values and game logic omitted), see Fig. 1 for its operation.

Fig. 1. Bubble shooter game with Cellulo platform [3] where the goal is *e.g.* to build the largest group of the same color. Launched robots collide elastically with existing robots (all 3 figures) and the walls (middle and right) before stopping.

M. Dorigo et al. (Eds.): ANTS 2018, LNCS 11172, pp. 427–428, 2018.
https://doi.org/10.1007/978-3-030-00533-7

```
Swarm{          //Runs physics simulation and other periodic updates of Robots
  Repeater{
    count: 20
    Robot{    //Imaginary robot (a point mass) that the physical robot follows
      color: "white"        //Property that updates LEDs of robot when changed
      GraspDetector{ id: graspDetector
        onGrasped: {      //Event that fires when the robot is touched by user
          var rand = Math.random();
          if(rand < 0.3333)       parent.color = "red";
          else if(rand < 0.6667) parent.color = "green";
          else                        parent.color = "blue";
        }
      }
      FixedToPhysicalRobot{      //Follows physical robot instead when enabled
        enabled: graspDetector.isGrasped
      }
      LaunchDetector{
        onLaunched: {                       //Fires when grasp is released by user
          parent.vel = launchVel;    //launchVel is mean vel. of last N frames
          dampingTimer.timedDisable();
        }
      }
      ViscousDamping{    //Force against and proportional to velocity, F = -cV
        coeff: 4.0                                    //Coefficient c in F = -cV
        Timer{ id: dampingTimer                            //Standard QML object
          interval: 5000                                        //In milliseconds
          onTriggered: parent.coeff = 4.0          //Fires when timer elapses
          function timedDisable(){
            parent.coeff = 0.0;
            restart();             //Starts timer, restarts if already running
          }
        }
      }
    }
  }           //For forces below, physical robot width is assumed to be 75 mm
  AlignmentAttraction{       //Aligns Robot pairs towards axes with force that
    dist: 150                //is orthogonal to pair, when closer than dist mm
    anglePeriod: Math.PI/3    //Aligns to 0, 60, 120, 180, 240 and 300 degrees
  }
  Attraction{ dist: 150 } //Pulls Robot pairs closer when closer than dist mm
  Repulsion{ dist: 100 }   //Pushes Robot pairs away when closer than dist mm
  BouncyContainer{            //Applies inwards force when container is exited
    rect: { x: 100, y: 100, w: 800, h: 800 }    //Whole arena is 1000x1000 mm
  }
}
```

The resulting programs are concise and contain no robotic implementation details, hiding what is uninteresting from the HCI perspective and exposing what is essential. However, our approach is centralized and cannot be readily applied to many existing platforms who do not guarantee global awareness. Moreover, developers must design and/or tune the desired forces, which may not be trivial.

Acknowledgments. Supported by the Swiss National Science Foundation through the National Centre of Competence in Research (NCCR) Robotics.

References

1. QML Applications. https://doc.qt.io/qt-5/qmlapplications.html. Accessed 28 June 2018
2. Ishii, H., Lakatos, D., Bonanni, L., Labrune, J.B.: Radical atoms: beyond tangible bits, toward transformable materials. Interactions **19**(1), 38–51 (2012)
3. Özgür, A., Lemaignan, S., Johal, W., Beltran, M., Briod, M., Pereyre, L., Mondada, F., Dillenbourg, P.: Cellulo: versatile handheld robots for education. In: ACM/IEEE International Conference on Human-Robot Interaction (2017)
4. Spears, W.M., Spears, D.F., Heil, R., Kerr, W., Hettiarachchi, S.: An overview of physicomimetics. In: Şahin, E., Spears, W.M. (eds.) SR 2004. LNCS, vol. 3342, pp. 84–97. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30552-1_8