Rahul Banerjee

CONTACT INFORMATION

Bill & Melinda Gates Center
Box 352355, Seattle WA 98195

Email: banerjee@cs.uw.edu

DISSERTATION

A Text-Free and Visually Concrete Programming Paradigm for Novices to Rapidly Create and Modify Graphical Interactive Programs

RESEARCH INTERESTS

Visual Progamming Languages, Computer Graphics, Human Computer Interaction

EDUCATION

The University of Washington, Seattle, WA — **Sep 2011 – Dec 2019**
Ph.D., *Computer Science* (Advisor: Zoran Popović) (expected)

The Indian Institute of Technology, Kanpur, India — **Jul 2003 – May 2005**
Master of Technology, *Computer Science*

University of Kalyani, Kalyani, India — **Aug 1999 – May 2003**
Bachelor of Technology, *Information Technology*

SELECTED PUBLICATIONS

Banerjee, Liu,, Sobel, Pitt, Lee, Wang, Chen, Davison, Yip, Ko, and Popović Empowering Families Facing English Literacy Challenges to Jointly Engage in Computer Programming, *CHI 2018*. [**Honorable Mention: Top 5% of 2500 submissions**]

Banerjee, Yip, Lee, and Popović. Empowering Children To Rapidly Author Games and Animations Without Writing Code, *Interaction Design and Children 2016*.

Tuite, **Banerjee**, Snavely, Popović, and Popović. PointCraft: Harnessing Players' FPS Skills to Interactively Trace Point Clouds in 3D, *FDG, 2015*. [**Best Paper**]

Banerjee and Mukherjee. Animating Hand Behaviours Using Virtual Sensors and an Automata Hierarchy, *ACIAR 2005*, May 11-13, 2005, Bangkok, Thailand.

RELEVANT PROJECTS

BlockStudio — **Sept 2013 – Present**
Creator, BDFL
A programming environment without textual code. Programming usually requires users to write abstract text to describe state changes in future scenarios.

By leveraging Programming By Demonstration, we can synthesize generic code in response to people modifying specific objects on the screen, within a concrete context. This empowers novices to rapidly create programs, by concretely enacting (instead of abstractly describing) what should happen. My research has shown that even people facing literacy challenges can learn BlockStudio and use it to make non-trivial programs.

I designed and implemented the language, its UI, its Programming-By-Demonstration inference algorithm, and I built the website where BlockStudio is freely accessible online. I also designed and built curriculum to teach this system and I manage/moderate the online community of BlockStudio users (mostly children aged 8-14).

Professional Experience

Pixar, Emeryville, CA **Jun 2012 – Sep 2012**
Research Intern, Studio Tools

- Physically-Based Illumination Inside Raytracer (C++): Running in realtime, for lighting artists to quickly preview illumination in complex scenes. Validated for pixel-exact match against equivalent RenderMan shader. This was built during **Monsters University** production, when Pixar started deploying real-time raytracing in artist workflows.

DreamWorks/Technicolor, Bangalore, India
Pipeline Technical Director, Character FX **Nov 2009 – Apr 2011**

- Realtime Procedural Fur Preview: Character FX artists would edit a few guide hairs, then render on the farm to see the resulting filled-in fur volume (turnaround time: 30 minutes). I wrote a custom Maya plugin (C++ and OpenGL) that generated and rendered a very close approximation of the fur volume, at interactive rates (30 fps). Written during **Puss in Boots** production, it was also used in subsequent feature-length animations, like **Madagascar 3**.
- Fur Collision Solver Parallelization: Characters with fur must have hairs bent out of the way so they don't protrude through clothes, belts, etc. This bending was computed one frame at a time, for frame-to-frame coherence, preventing parallelization. Since each individual fur hair computation is independent of other hairs, I carved it up via a "Map-Reduce" approach, implemented using available studio pipeline tools.
- Triage and Fix Problems: I supported 14 artists in the Character FX department, and solved their geometry/rendering/simulation issues. This required interfacing with other departments' technical staff (Animation, Lighting) and their pipelines.

NVIDIA, Bangalore, India **Jul 2005 – Nov 2009**
Systems Software Engineer, Embedded and Desktop

- Desktop GPU OpenCL 1.0 Compiler: I ported/implemented math routines (sin, cos, log, sqrt, rsqrt, etc.) to the internal GPU ISA (PTX). My implementations conformed to the error tolerances defined by the spec, and our compiler was the world's first GPU-based OpenCL 1.0 implementation to pass Khronos conformance.
- Automated Test Framework: I designed and built a system to allow every code checkin to trigger relevant test suites on all of our embedded platforms, in a completely automated fashion. This was logistically challenging, because most boards required manual login in over a serial console, hard reboots between tests, etc.
- Embedded GPU OpenGL Driver: I helped maintain Nvidia's OpenGL ES 1.x driver, improving coverage for new embedded platforms. I also wrote the blit engine and shader linker for the GeForce 8 OpenGL-ES 2.0 driver.
- Embedded Platform Support: I supported the cross-compiler toolchain for customers of Nvidia's ARM-based embedded platforms. This included building and maintaining binaries of gcc, g++, etc. for customer-specific platforms, as well as fixing bugs (or providing workarounds for bugs) in the above. I also wrote kernel drivers (e.g. to expose hardware registers for faster watchpoint support in gdb), and hand-coded assembly (e.g. to speed up memcpy on a platform which had a vanilla glibc implementation) as needed. This provided Nvidia's customers a powerful development environment on their desktops, using tools like Eclipse to efficiently step through code running on embedded boards.
- Video Encoding: I implemented H.264-compliant video deblocking on a custom Tensilica processor. I also constructed a transaction-level simulation of the encoding, to derive accurate estimates of processor cycles needed. This chip was used in a Motorola Razr phone, providing QVGA video encoding at 30 fps.

Languages

Haxe, Python, C++, Javascript/HTML/CSS, LaTeX